Universidade de Aveiro Escola Superior de Tecnologia e Gestão de Águeda

2015

Luís Carlos de Melo Goulart

Gestão de Trajetos Fora de Estrada

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Geoinformática, realizada sob a orientação científica do Doutor Mário Jorge Ferreira Rodrigues, Professor Adjunto da Escola Superior de Tecnologia e Gestão de Águeda da Universidade de Aveiro, e do Doutor Pedro Alexandre de Sousa Gonçalves, Professor Adjunto da Escola Superior de Tecnologia e Gestão de Águeda da Universidade de Aveiro

o júri

presidente	Prof. Doutor Joaquim José de Castro Ferreira Professor Adjunto da Escola Superior de Tecnologia e Gestão de Águeda da Universidade de Aveiro
vogal	Prof. Doutor Diogo Nuno Pereira Gomes Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro
vogal	Prof. Doutor Mário Jorge Ferreira Rodrigues Professor Adjunto da Escola Superior de Tecnologia e Gestão de Águeda da Universidade de Aveiro

agradecimentos

Os meus agradecimentos aos meus orientadores, Professor Mário Jorge Ferreira Rodrigues e Professor Pedro Alexandre de Sousa Gonçalves, pelo apoio e disponibilidade prestados ao longo do desenvolvimento da presente dissertação.
Agradecimentos à minha família, sobretudo aos meus pais, cujo esforço e dedicação e apoio contribuíram em grande medida para a conclusão da minha formação académica.
Finalmente, uma palavra de agradecimento aos colegas e docentes do Mestrado em Geoinformática, Licenciatura em Tecnologias de Informação, assim como colegas, docentes e não docentes da Escola Superior de Tecnologia Egestão de Águeda, em geral, com os quais convivi ao longo dos meus estudos, tendo proporcionado excelentes momentos de partilha de ideias e conhecimentos, cuja contribuição foi relevante no desenvolvimento da presente dissertação.

palavras-chave

Aplicação WebSIG, Web Mapping, Percursos Todo Terreno, Pesquisa de Percursos, FIWARE, Web Services.

resumo

Anualmente, realizam-se no país inúmeras iniciativas de Todo-Terreno Turístico (TTT) onde são automaticamente registadas as coordenadas de Global Positioning System (GPS) por aplicações de dispositivos móveis. Este tipo de informação pode ser utilizada, quer para fins de divulgação turística, quer por outro tipo de entidades que necessitem de circular nesses caminhos rurais, tipicamente no meio da montanha. Entre outras, são registadas a posição, velocidade e altitude do veículo, o que permite obter informações relevantes, tais como, se o percurso se encontra transitável ou qual a velocidade recomendada. Por exemplo, durante os combates a incêndios, os bombeiros e proteção civil poderão saber se estes percursos são utilizáveis no planeamento dos combates a incêndios com reduzida probabilidade de complicações relativa ao acesso dos veículos, melhorando assim o tempo de resposta.

O presente documento discute como poderá ser concebida uma aplicação web mapping, de código aberto, que permita a partilha, utilização e valorização de dados relativos aos percursos todo-terreno dos praticantes de TTT. O presente documento descreve como a aplicação desenvolvida no âmbito da dissertação de mestrado permite selecionar e ordenar possíveis trajetos que incluem os trajetos de TTT, apresentando as caraterísticas do terreno de modo a auxiliar a tomada de decisão por membros das corporações de Bombeiros. Será igualmente apresentada a interface atual da aplicação que inclui um mapa dinâmico e um gestor de pontos de referência.

keywords

WebGIS application, Web Mapping, Offroad Tracks, Path Search, FIWARE, Web Services.

abstract

Each year, several touristic off-road initiatives (TTT, from Portuguese Todo-Terreno Turístico) are organized, where Global Positioning System (GPS) track coordinates are recorded by participants, using handheld devices. This kind of information can be used either for tourism advertisement, or by other entities who need to use such roads, typically in rough terrains. Among other, vehicle position, speed, and altitude are usually registered. This data allows for inferring relevant information, such as the difficulty of the road and the recommended speed for a variety of scenarios. For instance, during fire-fighting, fire and civil protection can verify if these paths are useful and appropriate for their vehicles, reducing the likelihood of vehicle access complications, thus improving response time.

This dissertation discusses the design of an open source web mapping application that features track sharing and recovery of data on such tracks, in order to generate value for other activities. It describes how the application selects and sorts paths, composed by off-road tracks and regular roads, using diverse features to assist the decision making process of institutions, such as Fire Department corporations. The current interface of the application is presented, which includes a dynamic map and a landmark manager.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

1.1 Enquadramento

Os participantes de Todo-Terreno Turístico (TTT), por norma, gravam os percursos com recurso a aplicações *Global Positioning System* (GPS), que monitorizam informações como a posição, velocidade e altitude do veículo, ao longo do percurso. Essa informação é normalmente partilhada entre estes, para que se dê a conhecer um percurso efetuado, e permitir que outros participantes refaçam o mesmo percurso. Com a massificação dos dispositivos GPS, os ficheiros de percursos tem vindo a substituir os *road books*, trazendo vantagens em termos trabalho, e conforto do copiloto.

A distribuição destes ficheiros faz-se, atualmente, por meio dos web sites dos clubes de TTT, por email, ou com recurso a sites de partilha de percursos de aventura, onde são colocados juntamente com percursos de BTT, caminhada, escalada, etc. Esta alternativa não é vocacionada para a prática de TTT, em particular.

Anualmente, realizam-se no país inúmeras iniciativas TTT, sendo efetuados levantamentos de percursos. Essa informação permite tomar conhecimento do estado dos percursos, como por exemplo, se este se encontra ou não transitável, ou até mesmo saber a velocidade recomendada para cada percurso.

Os percursos são geralmente realizados por caminhos de montanha, que frequentemente ficam intransitáveis, devido à queda de árvores, muros de suporte, entre as mais diversas causas.

A prática de TTT encontra-se interdita entre os meses de Julho e Setembro, no entanto, a informação sobre percursos poderá ser útil para corporações de bombeiros e proteção civil. Durante os combates a incêndios, estas entidades recorrem, com frequência, a estes percursos. Sendo assim, o conhecimento do estado dos mesmos é de grande importância, pois deverá facilitar o planeamento dos combates a incêndios, e ao mesmo tempo, reduzir a probabilidade de complicações, relativas ao acesso dos veículos.

1.2 Objetivos

O presente documento reporta o desenvolvimento de uma aplicação do tipo *web mapping*, de código aberto, que permite a partilha de dados relativos aos percursos todo-terreno, por parte dos membros de clubes Todo-Terreno Turístico (TTT), e que essa mesma informação possa ser acedida, em conjunto com funcionalidades que auxiliem a tomada de decisão, por membros das equipas de socorro, como por exemplo, Bombeiros, Polícias ou Proteção Civil.

A aplicação desenvolvida é direcionada à utilização por parte de utilizadores aficionados pelo TTT, e entidades associadas ao socorro. Trata-se uma aplicação do tipo cliente-servidor,

acessível pela Web, em que o acesso aos dados, e respetivo processamento, realizam-se por intermédio de *Web Services*. De forma a satisfazer as necessidades dos utilizadores ligados ligados aos clubes TTT, a aplicação proporciona as seguintes funcionalidades:

- Inserção de percursos, a partir de ficheiros em formato GPX.
- Inserção de Pontos de Interesse (POI), em que estes correspondem a informações adicionais, de caráter pontual, sobre o estado dos percursos, tais como obstruções, derrocadas, zonas onde é possível realizar inversão de marcha, curvas apertadas, entre outros.
- Criação de segmentos de percurso, contendo informações adicionais sobre determinadas porções dos respetivos percursos, tais como o tipo de piso, ou a largura do segmento de percurso.
- Pesquisa de percursos, com base em múltiplos parâmetros, capaz de retornar resultados de caráter compensatório e não compensatório, e disponibilização dos mesmos sobre um mapa.
- Descarregamento dos percursos correspondentes aos resultados de uma pesquisa.

1.3 Metodologia

O trabalho efetuado no âmbito do desenvolvimento da aplicação proposta no presente documento distribuiu-se por algumas etapas distintas. A primeira consistiu na especificação de funcionalidades a implementar, e definição dos recursos de software e hardware necessários para o desenvolvimento de uma aplicação Web.

A etapa seguinte envolveu uma investigação inicial sobre o estado da arte relativamente a aplicações Web equiparadas, assim como tecnologias, plataformas, bibliotecas e standards utilizados na atualidade, no desenvolvimento deste tipo de aplicações. Esta investigação resultou na tomada de decisão relativamente às tecnologias a utilizar no desenvolvimento da aplicação proposta.

Procedeu-se, de seguida ao desenvolvimento da aplicação. Nesta etapa, algumas funcionalidades foram definidas como prioritárias, estabelecendo-se assim uma ordem de implementação, tendo em vista, também como prioridade estabelecida, o aspeto funcional, relegando-se a otimização para segundo plano. Ainda nesta etapa, algumas funcionalidades foram remodeladas, em relação às especificações iniciais.

A última etapa correspondeu à produção do presente documento, o qual documenta as etapas acima descritas, passando por uma análise aos resultados dos testes efetuados, e terminando com as respetivas conclusões, e propostas de implementação das funcionalidades deixadas em aberto.

1.4 Contribuições

No decorrer do trabalho relatado no presente documento, destacam-se três contribuições para a comunidade:

- **Estudo de uma solução** - Foi realizado um estudo sobre as tecnologias, plataformas e bibliotecas de código aberto, existentes atualmente, aplicáveis no desenvolvimento da aplicação descrita no presente documento. As mesmas são abordadas ao longo dos restantes capítulos deste documento.

- **Artigo** - A aplicação descrita neste documento foi proposta num artigo apresentado na 10ª Conferência Ibérica de Sistemas e Tecnologias de Informação, realizada em Águeda, Aveiro, Portugal, entre os dias 17 e 20 de Junho de 2005. O artigo intitulado "Percursos Todo Terreno Turísticos - Proposta de uma aplicação Web Mapping para partilha e pesquisa de percursos" expõe uma fase intermédia do desenvolvimento do trabalho.

- **Implementação de um protótipo** - Foi desenvolvido o protótipo de uma aplicação do tipo WebSIG, a qual tem como objetivo principal a partilha e pesquisa de percursos TTT. Embora possa ser utilizada por qualquer pessoa, a aplicação tem dois tipos particulares utilizadores alvo. os praticantes de TTT, funcionando como uma plataforma onde podem inserir o traçado GPS dos percursos por estes efetuados, partilhando assim com a restante comunidade. As entidades ligadas ao socorro correspondem ao outro tipo, visto que podem pesquisar a informação recolhida pelos praticantes TTT, para planear as atividades de salvamento, combate a incêndios, entre as mais diversas.

1.5 Estrutura do Documento

O capítulo 2 relata o estado da arte no que respeita às aplicações WebSIG, e ao respetivo desenvolvimento das mesmas. São abordadas as tecnologias aplicáveis no desenvolvimento, assim como alguns trabalhos recentes, relacionados com o tema.

O capítulo 3 proporciona uma descrição das funcionalidades efetiva ou parcialmente implementadas na aplicação. Trata-se de uma descrição de caráter técnico, que visa dar a entender as sequências de eventos relacionadas com cada funcionalidade, e de que forma processam a informação, até se obter os resultados pretendidos. Apenas as funcionalidades principais são abordadas, como por exemplo a pesquisa de percursos, relegando aspetos particulares da implementação, como por exemplo a disponibilização de cartografia base, para o capítulo 4.

Conforme referido anteriormente, o capítulo 4 aborda os aspetos particulares da implementação, no entanto, proporciona uma descrição do processo de desenvolvimento de algumas das funcionalidades, revelando dificuldades encontradas, e soluções encontradas para as mesmas.

O capítulo 5 descreve os testes efetuados às funcionalidades, demonstra os respetivos resultados, seguidos de uma breve reflexão sobre os mesmos.

No capítulo 6 são descritas as conclusões a que se chegou, não só a partir dos resultados, como também retiradas ao longo do processo de desenvolvimento. É feita uma reflexão final sobre as funcionalidades implementadas, e ficam propostas sugestões de implementação para as funcionalidades em aberto.

Capítulo 2

Estado da Arte

O presente capítulo tem como objetivo mostrar o atual estado da arte no que respeita ao desenvolvimento de aplicações *Web Mapping*, ou *WebSIG*. Começa-se por abordar as tecnologias e normas aplicáveis no desenvolvimento deste tipo de aplicações, terminando numa abordagem a alguns trabalhos relacionados com a temática da aplicação desenvolvida, elaborados recentemente.

2.1 Sistemas de Coordenadas Espaciais

A propriedade básica dos sistemas de informação geográfica, em contraste com outros sistemas de informação, consiste no facto de os dados armazenados encontrarem-se georreferenciados. Tal significa que os mesmos tem a respetiva localização definida na superfície terrestre, através de coordenadas definidas por um sistema de coordenadas georreferenciado. A superfície terrestre apresenta uma forma complexa, aproximadamente esférica, razão pela qual a definição de um sistema de coordenadas apropriado torna-se relativamente complexa. Um sistema de coordenadas pode ser definido de duas formas distintas. Pode ser definido com base numa esfera, ou elipsoide, originando um sistema de coordenadas geográfico, ou, em alternativa, a esfera é projetada numa superfície passível de ser transformada num plano, onde pode ser definido um sistema de coordenadas cartesiano [3].

Serão abordados, de seguida, dois sistemas de coordenadas, considerados relevantes para a aplicação desenvolvida no âmbito da dissertação descrita no presente documento, nomeadamente, *World Geodetic System - 1984* (WGS84), e *European Terrestrial Reference System - 1989* (ETRS89).

O WGS84 corresponde a um sistema geodésico global, definido pela *United States Defense Mapping Agency* (atualmente designada por NIMA), o qual é utilizado pelo sistema GPS, pelo que os recetores GPS determinam coordenadas com base neste sistema. Encontra-se associado a este sistema de coordenadas um elipsoide equipotencial de revolução, cujo centro coincide com o centro de massa da Terra, assim como uma projeção cartográfica Transversa de Mercator. [4]. Este sistema é materializado por um conjunto de estações designadas por *International GPS Service for Geodynamics* (IGS), as quais encontram-se distribuídas por todo o planeta [5].

Devido a efeitos de geodinâmica, as coordenadas variam anualmente, na ordem do centímetro, razão pela qual é habitual fixar o sistema WGS84 numa determinada época [5].

À semelhança do WGS84, o sistema ETRS89 corresponde também a um sistema global, cujo elipsoide associado apresenta características semelhantes, assim como a mesma projeção cartográfica. Criado pela subcomissão europeia *European Reference Frame* (EUREF) da Associação Internacional de Geodesia para o Sistema de Referência Europeu [4]. Este sis-

tema coincide com o WGS84, à data de 1989. É materializado por um conjunto de estações localizadas na placa euro-asiática [5].

2.2 Bases de Dados Espaciais

As bases de dados espaciais armazenam dados geográficos num sistema de ficheiros adequado para conjuntos de dados de grandes dimensões, podendo este conter milhares de características, e providenciam um mecanismo eficiente para armazenamento, interrogação, análise e atualização dos referidos dados. A maioria das bases de dados espaciais são extensões de bases de dados baseadas em *Structured Query Language* (SQL) existentes, que implementam o padrão *Open Geospatial Consortium - Simple Feature Access* (OGC-SFS) para SQL, o qual define de que forma os objetos espaciais devem ser representados [6].

As seguintes subsecções abordam em maior detalhe algumas das bases de dados espaciais existentes na atualidade.

2.2.1 PostGIS

O *PostGIS* foi desenvolvido como uma extensão espacial para a base de dados *PostgreSQL*. Considera-se atualmente que esta extensão providencia a mais extensa implementação do padrão OGC-SFS. Adicionalmente, apresenta um vasto suporte para dados *raster*, e respetiva análise, para os quais incorpora a biblioteca *Geospatial Data Abstraction Library* (GDAL), proporcionando suporte a um vasto conjunto de formatos *raster*. A extensão disponibiliza três novos tipos de dados, designadamente *geometry*, *geography* e *raster*, e suporta esquemas de indexação espacial, para rápida obtenção de dados. Inclui ainda uma vasta biblioteca de funções para análise de objetos vetoriais ou *raster*, conversões entre vetores e *rasters* e transformações entre sistemas de referência espaciais [6].

2.2.2 SpatiaLite

O *SpatiaLite* é uma extensão espacial para a base de dados *SQLite*. Este projeto tem em vista ser aproximadamente equivalente ao *PostGIS*, mas com características bastante mais ligeiras em termos de consumo de recursos computacionais. Utiliza a biblioteca de geometrias de GEOS para implementar o padrão OGC-SFS. Inclui uma biblioteca de funções com dimensões semelhantes da biblioteca *PostGIS*, no entanto, estas assumem geometrias planares, e ignoram efetivamente o sistema de referência espacial dos dados.

A base de dados *SpatiaLite* funciona bem em ambientes de utilizador único, mas não se encontra bem preparada para lidar com múltiplas ligações concorrentes, como acontece frequentemente em ambiente Web [6].

2.2.3 MySQL spatial

O *MySQL spatial* consiste numa implementação básica do padrão OGC-SFS. Suporta formatos vetoriais, no entanto, atualmente não suporta formatos *raster*. Providencia uma biblioteca de funções bastante mais reduzida que as bases de dados espaciais descritas anteriormente. Todos os cálculos assumem geometrias planares. Os tipos de dados e funções espaciais podem ser utilizados em diversos mecanismos de armazenamento do *MySQL*, incluindo *MyISAM*, *InnoDB* e *ARCHIVE*, existindo suporte para indexação espacial em tabelas *InnoDB* e *MyISAM* [6].

2.3 Difusão de Dados Espaciais através de Web Services

Os dados espaciais podem ser difundidos com recurso a servidores de dados espaciais. Estes assumem o papel de disponibilizar a informação geográfica em formatos *Web-friendly*, por intermédio de *Web Services* padrão da OGC, a qual pode, posteriormente, ser renderizada numa página Web visualizada num *browser*, com recurso a uma biblioteca ou extensão para mapeamento [6].

Nos últimos anos, a OGC, que tem vindo a trabalhar no desenvolvimento de especificações para partilha e interoperabilidade de informação geográfica e definiu um conjunto de especificações de *Web Services* para informação geográfica num *framework* designado por *OGC Web Services* (OWS). Estas especificações têm vindo a ser suportadas por diversas plataformas SIG de código aberto, e muito usadas na prática [7], tais como *Geoserever*, *MapServer*, entre outras.

O *framework* OWS contém diversas especificações de serviços, interfaces e padrões, entre os quais, *Web Map Service* (WMS), *Web Feature Service* (WFS), *Web Coverage Service* (WCS) e *Web Processing Service* (WPS), entre outros [7].

A norma WMS fornece uma interface baseada em HTTP para efetuar pedidos de imagens mapa a partir de uma ou mais bases e dados espaciais distribuídas. O utilizador ao efetuar um pedido a este serviço especifica os elementos de interesse, como por exemplo uma área, tendo como resposta uma imagem mapa referente a essa área, em formato imagem. Consideram-se duas categorias deste serviço: *Basic* WMS, que se limita a retornar a referida imagem, e *Queryable* WMS, que disponibiliza informações extra sobre a *feature* representada na imagem [8].

A norma WFS fornece uma interface para aceder e manipular *features* geográficas. Permite operações de *Insert*, *Update*, *Delete* ou descoberta de *features* geográficas expressas em *Geography Markup Language* (GML). O serviço WFS pode estar incluído em duas categorias, *Standard* WFS, ou só de leitura, que apenas implementa operações relacionadas com a obtenção de dados, e *Transactional* WFS, que implementa a operação de transação, e que permite a modificação dos dados [8].

A norma WCS define uma interface para intercâmbio de informação geográfica em formato de *coverage*. Neste formato, os dados podem variar no espaço e no tempo. Através de um serviço WCS, um utilizador tem a possibilidade de especificar os critérios para as *queries*. A resposta poderá ser, dependendo da operação efetuada, a descrição das *coverages* pretendidas, ou uma determinada *coverage*, propriamente dita [8].

A norma WPS foi especifica a forma como os algoritmos SIG são disponibilizados pela Internet. Proporciona uma solução para um cliente efetuar o pedido de uma operação espacial a partir de um serviço. Tem em vista a automação do geo-processamento ao empregar semântica geo-espacial numa arquitetura baseada em serviços [9].

A norma *Web Map Tile Service* (WTMS) define uma interface para servir mapas digitais com recurso a *tiles* de imagens pré-definidos. Complementa o padrão WMS, permitindo resolver um problema de flexibilidade existente nos serviços WMS. A capacidade de processamento no servidor deve escalar com o número de clientes ligados, no entanto, o potencial de *cache* para imagens a servir é limitado. A estratégia definida por esta norma passa pela utilização de *tiles* pré-renderizadas de imagens, armazenadas localmente, que não necessitem de posterior manipulação ou geo-processamento. Os clientes, sempre que necessitem de uma nova imagem, limitam-se a pedir o conjunto de *tiles* em falta, relativamente aos que já possuíam previamente, encarregando-se eles próprios da criação da imagem final, a partir do respetivo conjunto de *tiles* [10].

A norma *Catalogue Services for the Web* (CSW) especifica interfaces, ligações e um *fra-*

mework para a definição de perfis necessários para a publicação e acesso a catálogos digitais de meta-dados para dados espaciais, serviços e informação relacionada [11].

As seguintes subsecções abordam algumas das soluções existentes, na atualidade, para a disponibilização de *Web Services*.

2.3.1 Geoserver

O *Geoserver* é uma aplicação de código aberto destinada à manipulação e disseminação de dados geoespaciais. Providencia as funcionalidades básicas para a criação de Infraestruturas de Dados Espaciais (IDE), de acordo com as normas definidas pelas organizações (OGC) e *ISO Technical Committe 211* (ISO TC 211). O *Geoserver* foi criado para consumir, gerir e servir dados geoespaciais, do tipo vetorial, e *raster*, assim como para criar e disseminar mapas georreferenciados, obtidos pela sobreposição de versões renderizadas dos referidos dados, de acordo com com regras de estilos implementadas segundo normas específicas [12].

Trata-se de uma aplicação desenvolvida em *Java*, podendo assim ser utilizada na maioria dos sistemas operativos, disponibilizada em formato *web archive* (WAR), funcionando, desta forma, sobre aplicações *servlet container*, como por exemplo, *Apache Tomcat* ou *Jetty*. Proporciona uma interface gráfica de administração, para configuração, no entanto, pode também ser configurado através de uma interface do tipo *Representational State Transfer* (REST) [6].

O *Geoserver* implementa os *web services* padrão da OGC, designadamente WMS, WFS, WCS e WPS. Outras funcionalidades incluem suporte integrado ds bibliotecas de *mapping* *OpenLayers* e *Google Earth*, *caching* espacial automatizado, nomeadamente *GeoWebCache*, e um amplo suporte para bases de dados espaciais, como por exemplo, *PostGIS*, *ArcSDE*, *Oracle* e *DB2*. O *Geoserver* depende em grande medida da biblioteca *GeoTools*, uma biblioteca desenvolvida em *Java*, de código aberto, que disponibiliza suporte SIG para tipos de dados espaciais, nomeadamente camadas vetoriais e *raster* [6].

2.3.2 MapServer

O *MapServer* é uma aplicação do tipo *Common Gateway Interface* (CGI), desenvolvida em linguagem C, que pode ser instalada em qualquer sistema operativo. A implementação em C proporciona a esta aplicação um desempenho superior, relativamente a aplicações equiparadas, desenvolvidas em *Java*. Disponibiliza os *Web Services* padrão da OGC, designadamente WMS, WFS e WCS. É configurado através de ficheiros específicos, designados por *Mapfiles*. Possui uma API designada por *MapScript*, que pode ser utilizada para configurar o servidor, assim como para manipular ou interagir com os dados armazenados. Esta API está disponível para diversas linguagens de programação, incluindo *Python*, *Java* e PHP. Os *datasets* servidos pelo *MapServer* podem ser armazenados no sistema de ficheiros do servidor, ou em bases de dados com extensão espacial [6].

2.3.3 deegree

O *deegree* consiste numa implementação, em *Java*, dos *Web Services* padrão da OGC, que funciona em todos os sistemas operativos. Disponibiliza implementações dos padrões WFS, WMS, CSW, WMTS e WPS. Assim como o *Geoserver*, o *deegree* disponibiliza uma interface Web de configuração, assim como uma interface REST [6].

2.4 Bibliotecas de Mapeamento

As bibliotecas de mapeamento são necessárias para visualizar informação espacial em ambiente Web. Estas consomem dados disponibilizados a partir dos *Web Services* padrão, e renderizam os mapas para apresentação num cliente, como por exemplo, um *web browser* [6].

As seguintes subsecções abordam, em maior detalhe, algumas das bibliotecas de mapeamento disponíveis atualmente para a renderização de informação geográfica em *web browsers*.

2.4.1 OpenLayers

O *OpenLayers* é biblioteca desenvolvida inteiramente em *javascript*, destinada à construção de aplicações geoespaciais baseadas em Web, similares ao *Google Maps*. Permite renderizar dados vetoriais ou *raster* a partir de uma variedade de formatos, incluindo GeoJSON, KML, GML, assim como *Web Services* padrão da OGC. Além disso, disponibiliza métodos para desenhar sobre o mapa, e editar dados interativamente. Permite ainda utilizar uma variedade de serviços de cartografia base, nomeadamente *Open Street Map* (OSM), *Bing*, *MapQuest* e *Google*. Não requer extensões, nem tem restrições de utilização, no entanto, a utilização de cartografia base proprietária, como *Google* ou *Bing*, por exemplo, poderá estar sujeita a restrições de licença [6].

A nível conceptual, esta biblioteca tem em vista a separação das ferramentas de manipulação e visualização espacial, dos dados espaciais, tratando-se de uma ideia não implementada nas alternativas comerciais equivalentes [13].

2.4.2 GeoExt

O *GeoExt* corresponde a um projeto que combina *OpenLayers* com *Ext JS* para proporcionar um *framework* capaz de desenvolver aplicações WebSIG com aparência *desktop*. *Ext JS* é uma biblioteca *javascript* destinada à construção de interfaces Web, a qual inclui um conjunto de ferramentas de interface de utilizador, um modelo de componentes extensível, e uma API. Esta biblioteca, assim como os produtos relacionados com a mesma, tem experimentado sucesso recentemente, ganhando relevância em aplicações Web de elevado grau de complexidade. Esta biblioteca é atualmente utilizada por muitas das maiores organizações nos respetivos *websites*. Pode funcionar com uma biblioteca *standalone*, e pode ser integrada com a biblioteca *javascript Prototype* [13].

2.4.3 Google Maps

O *Google Maps* implementa um ambiente de mapeamento em 2D, com cartografia base de alta resolução. Permite o desenho de formas sobre o mapa, assim como a edição interativa de dados. Os dados são disponibilizados no formato KML, ou em alternativa, adicionados com recurso à API *javascript* [6].

2.4.4 Leaflet

O *Leaflet* é uma biblioteca de mapeamento *javascript*, de código aberto, a qual tem apresentado popularidade crescente, devido ao seu reduzido volume. Permite disponibilizar cartografia base a partir de diversos serviços de *tiles*, assim como a representação de *features* vetoriais sobre o mapa com recurso ao padrão *Scalable Vector Graphics* (SVG). Está integrada com *Mapbox.js*, o que permite o carregamento e a manipulação de forma simples, de conjuntos de *tiles* personalizados. Devido ao seu reduzido volume, assim como suporte a interações baseadas em toque, é considerada uma das melhores bibliotecas de mapeamento para

dispositivos móveis. Pode ser estendida através de *plugins*, também estes de código aberto [14].

2.5 Bibliotecas Espaciais

A presente secção descreve um conjunto de bibliotecas que fornecem um conjunto de componentes de software espaciais, que podem ser explorados tanto a nível *desktop*, com em aplicações WebSIG. Existem diversas bibliotecas espaciais, de código aberto, disponíveis online [6].

As seguintes subsecções abordam em maior detalhe algumas destas bibliotecas.

2.5.1 GDAL

Geospatial Data Abstraction Library (GDAL) corresponde a uma biblioteca de código aberto destinada à interpretação de formatos de dados espaciais, tanto vetoriais, como *raster* [15]. Disponibiliza um modelo de dados abstrato e unificado, para todos os formatos de dados suportados. Disponibiliza ainda, uma gama de ferramentas, executáveis em linha de comandos, para conversão e processamento de dados. Esta biblioteca encontra-se dividida em duas componentes distintas, consistindo na parte *raster*, e na parte vetorial, designada por OGR, em que cada uma delas disponibiliza um modelo de dados e API específicos [16]. Após a versão 2.0, ambas as componentes encontram-se integradas [15].

2.5.2 Proj.4

O *Proj.4* consiste numa biblioteca, de código aberto, para projeções cartográficas, amplamente utilizada em aplicações SIG, tanto na vertente cliente, como na vertente servidor, destacando-se como exemplo, o servidor de dados espaciais *MapServer* [6].

Permite converter coordenadas geográficas, ou seja, latitudes e longitudes, em coordenadas cartesianas, por intermédio de uma vasta gama de funções de projeção. A conversão de coordenadas cartesianas para coordenadas geográficas também é possível [17].

2.5.3 GeoTools

GeoTools é uma biblioteca de código aberto, desenvolvida em *Java*, que providencia funcionalidades SIG avançadas. Suporta formatos geo-espaciais vetoriais e *raster*, acesso a Sistemas de Gestão de Bases de Dados (SGBD), e renderização de mapas complexos. Trata-se de um dos mais antigos projetos da *Open Geospatial Foundation*, razão pela qual apresenta total conformidade com as especificações da OGC, incluindo GML, WMS, WFS, *grid coverage*, transformação de coordenadas e *Styled Layer Descriptor* (SLD). Pode ser utilizada tanto em vertente cliente, como em vertente servidor. O *GeoTools* é atualmente utilizado pelo *Geoserver* [6].

2.5.4 JTS

O *Java Topology Suite* (JTS) é uma biblioteca desenvolvida em *Java*, amplamente utilizada, para lidar com operações geométricas 2D. Apresenta conformidade com o modelo de geometria e API definidos pelo padrão OGC-SFS. Proporciona acesso a funções simples, tais como *buffering* e sobreposições, assim como funcionalidades mais avançadas, tais como algoritmos espaciais e estrutura de suporte, nomeadamente, indexação espacial, simplificação de geometrias, triangulação de *Delaunay*, entre outras. Esta biblioteca é utilizada em vários projetos geo-espaciais, incluindo *GeoTools*, *PostGIS* e *Geoserver* [6].

2.5.5 GEOS

O *Geometry Engine Open Source* (GEOS) corresponde a uma implementação em C++ da biblioteca JTS. Assim sendo, tem por objetivo disponibilizar todas as funcionalidades incluidas em JTS, incluindo as funções e operadores espaciais definidos pelo padrão OGC-SFS, assim como funções de topologia específicas de JTS [18].

2.6 Plataforma FIWARE

A plataforma *FIWARE* consiste numa plataforma que disponibiliza uma infraestrutura para a criação e distribuição de serviços digitais, de código aberto, que se baseia em elementos designados por *Generic Enablers* (GE), que oferecem funções reutilizáveis e vulgarmente partilhadas, que servem uma multitude de áreas de aplicação ao longo de vários sectores. Os GE distinguem-se pela capacidade de servir uma multitude de áreas de aplicação, ao passo daquilo que pode ser referido como *Domain-specific Common Enablers*, ou *Specific Enablers* (SE). Os segundos são comuns a múltiplas aplicações, mas todos eles são específicos a um conjunto muito limitado de áreas de aplicação [19].

Os objetivos chave do projeto FIWARE correspondem à identificação e especificação dos GE, juntamente com o desenvolvimento e demonstração de implementações de referência de GE identificados. Qualquer implementação de um GE compreende um conjunto de componentes e irá oferecer capacidades e funcionalidades, que podem ser personalizadas de forma flexível, utilizadas e combinadas para diversas áreas de aplicação distintas, permitindo o desenvolvimento de aplicações e serviços de Internet avançados e inovadores [19].

As especificações das APIs e protocolos de interoperabilidade suportados pelos GE são públicas e livres. Estas contém toda a informação necessária para criar produtos complacentes, que funcionem como implementações alternativas dos GE [19].

Dos vários GE disponíveis, será destacado o *POI Data Provider* (POI-DP), em virtude de ter sido utilizado numa parte nuclear da aplicação proposta no presente documento.

2.6.1 POI Data Provider

Os POI-DP correspondem a um GE que proporciona serviços de pesquisa espacial e dados, relativos a pontos de interesse (POI), através de uma API RESTful, disponibilizada por serviço web. O modelo de dados para a representação de POIs é modular, e baseado no modelo de dados entidade-componente. No modelo proposto, cada POI é uma entidade, a qual é identificada por um identificador único universal (UUID). Os dados relacionados com uma entidade são armazenados em diferentes componentes de dados, que são ligados à entidade através do UUID. Estes componentes de dados são independentes entre si, e cada entidade POI poderá ter um conjunto distinto de componentes. Isto implementa o conceito de servir uma vasta gama de grupos de interesse, uma vez que o modelo de dados pode ser facilmente estendido por componentes de dados de aplicações. Dados de POIs podem ser distribuídos, e diferentes componentes de dados podem ser administrados por diferentes organizações, utilizando diferentes políticas [1].

Modelo de dados para POIs

Diferentes organizações poderão disponibilizar diferentes tipos de informação sobre POIs, ou informações sobre diferentes POIs, por este facto seria muito difícil satisfazer todas as necessidades numa única estrutura de dados unificada [20].

Por esta razão, existe uma estrutura de dados modular, que suporta componentes independentes. O GE providencia os componentes nucleares, que contém informações básicas sobre os POIs, que permitam pesquisa espacial. Componentes de dados adicionais são posteriormente especificados, estendendo o modelo de dados de POIs, incluindo conteúdo 3D, itens multimédia e informações de sensores [20]. A figura 2.1 representa a arquitetura do serviço POI-DP, implementada por este GE, na qual é visível a referida estrutura de dados modular.

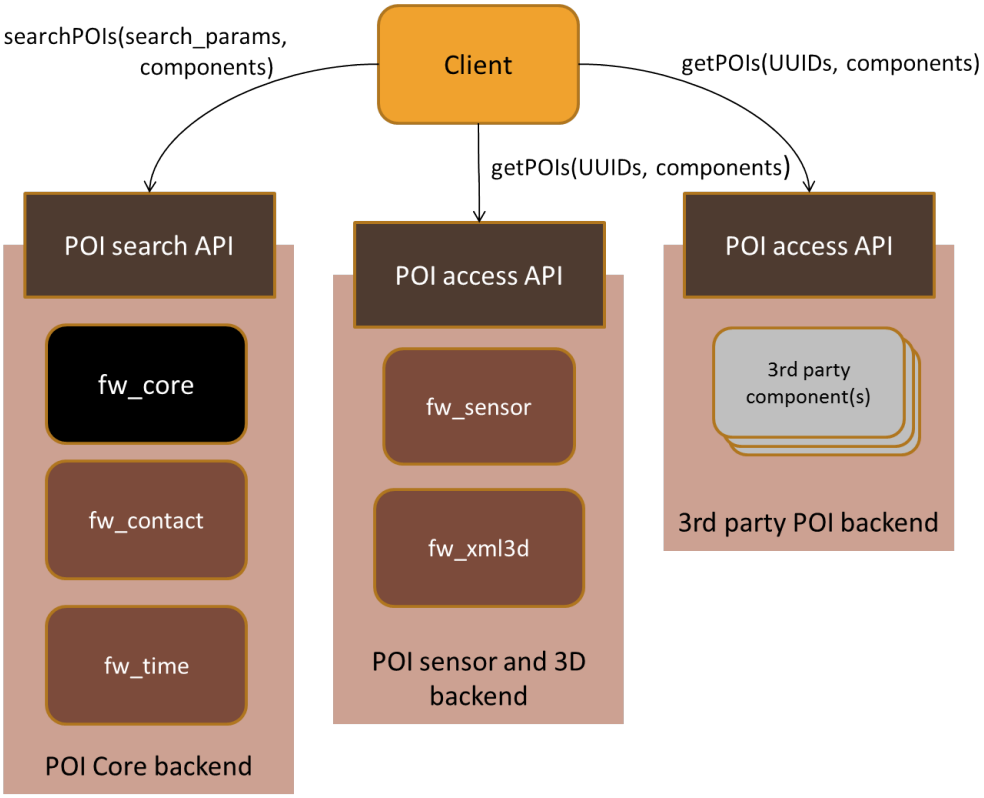

Figura 2.1: Arquitetura do serviço POI-DP, e modelo de dados para POIs [1][2]

O desenho deste modelo é baseado no princípio do modelo entidade-componente, que permite particionamento dos dados de POIs em componentes separados, que ficam ligados a uma entidade POI. Um único POI corresponde a uma entidade, que é identificada por um UUID. Os dados relacionados com uma entidade são armazenados em diferentes componentes, que se encontram ligados à entidade através da UUID. Estes componentes de dados são independentes uns dos outros, e o conjunto de componentes pode variar para cada POI. No entanto, todos os POIs devem ter um componente em particular, designado por *fw_ core*, que contém os dados nucleares mínimos que descrevem o POI [20].

Formato de representação

Os componentes de dados de um POI são representados como objetos JSON. A chave de cada objeto JSON identifica o tipo de componente de dados do POI que representa [20].

O objeto JSON de raiz, designado por "pois", contém todos os POIs que correspondem a uma determinada interrogação. Cada POI é representado como um objeto JSON, onde o UUID do POI corresponde à chave do objeto. Cada POI contém um conjunto individual de componentes de dados [20].

Serviços disponibilizados pelo POI-DP

A API deste GE permite aceder aos seguintes serviços, previamente implementados, de pesquisa e modificação de POIs [20]:

- **Pesquisa** - Serviços acedidos através de um pedido HTTP GET
  - **Pesquisa radial** - O caso de utilização mais comum é pesquisar POIs próximos de uma determinada localização. Este tipo de pesquisa pode ser utilizado para encontrar POIs em torno de uma determinada localização, contidos dentro de uma certa distância desta. A localização é dada em latitude e longitude, e o raio em metros.
  - **Pesquisa por *bounding box*** - Permite efetuar uma pesquisa de POIs no interior de uma dada *bounding box* Entende-se por *bounding box*, uma área definida por duas latitudes e duas longitudes [21].
  - **Acesso a POIs por UUID** - Para o caso de que o cliente conheça à partida os UUIDs dos POIs que lhe interessem. Neste caso, os dados referentes aos POIs são acedidos diretamente. Vários UUIDs podem ser retornados por uma única interrogação.
- **Modificação**
  - **Adicionar e modificar POIs** - Os dados do POI adicionado ou modificado são enviados pelo cliente com recurso a um pedido HTTP POST.
  - **Eliminar POIs** - Neste caso, recorre se a um pedido HTTP DELETE.

2.7 OpenStreetMaps

Existem atualmente diversos projetos de mapeamento gerado por utilizadores, a partir de informação recolhida por dispositivos GPS, dos quais, o projeto OpenStreetMap (OSM), é provavelmente o mais extenso e eficaz, correntemente em desenvolvimento. A existência destes projetos deve-se, sobretudo, a alguns fatores determinantes, ocorridos nos últimos anos. O primeiro corresponde a uma decisão tomada pelo Presidente dos Estados Unidos da América, Bill Clinton, anunciada no dia 1 de Maio de 2000, em que ficou definida a remoção da disponibilidade seletiva do sinal GPS, permitindo assim um considerável aumento de precisão dos recetores GPS de baixo custo. O segundo, ocorrido em 2002, foi a publicação do *standard* de intercâmbio de dados GPS eXchange (GPX, o qual foi rapidamente adotado pela maioria dos fabricantes de recetores GPS, facilitando, em grande medida, a manipulação e partilha de dados. O terceiro, teve a ver com a redução do preço de mercado dos recetores GPS, permitiu a cada vez mais pessoas recolher e partilhar informação geográfica com facilidade [22].

A motivação chave do projeto OSM passa por disponibilizar acesso livre a informação geográfica atualizada, onde a informação geográfica digital de relativa precisão é considerada dispendiosa, e de difícil acesso para indivíduos, ou pequenas empresas e organizações [22].

Atualmente, centenas de milhar de utilizadores contribuem para o projeto, a nível mundial. Diversas aplicações baseadas na informação geográfica disponibilizada pelo OSM foram desenvolvidas. Para além de mapas com temáticas distintas, a informação do OSM demonstra também potencial para aplicações mais avançadas, nomeadamente, *Location-based services* (LBS) ou serviços Web 3D. Além do mais, a implementação de dados OSM para áreas interiores, terá também sido discutida [23].

O cerne deste projeto está localizado no seu respetivo web site (www.openstreetmap.org), e é composto por quatro partes distintas. Em primeiro lugar, a interface de mapeamento OSM,

que permite aos utilizadores pesquisar o mapa do mundo, e descobrir quais as áreas geográficas que se encontram completas. É possível, ainda, descarregar porções de informação geográfica, em formato vetor ou *raster*. A interface de edição permite a qualquer pessoa contribuir para o projeto, digitalizando objetos geográficos, carregando ficheiros GPX com leituras de recetores GPS, ou corrigindo erros que eventualmente ocorram em edições anteriores. Por fim, a *OSM community wiki*, que contém informação sobre o projeto, disponibiliza orientação sobre boas práticas de mapeamento, para contribuintes básicos ou avançados, e documentação extensiva sobre a infraestrutura técnica do projeto. Apenas utilizadores registados podem realizar tarefas de edição, para que seja possível rastrear a origem da informação inserida, garantindo assim a manutenção da mesma, e prevenindo problemas relacionados com direitos de autor [22].

A informação armazenada na base de dados OSM é servida em conjuntos de *tiles*, entenda-se, conjuntos de imagens disponibilizadas em mosaico. Sempre que é alterada a vista do mapa, ou o nível de zoom, é gerado um novo conjunto de *tiles*, que é servido ao cliente de forma assíncrona. A renderização das *tiles*, para o caso do específico do web site do projeto OSM, são renderizadas com recurso à biblioteca *Mapnik*, de código aberto, destinada à criação de imagens de mapa, de alta qualidade. A renderização de *tiles* representa um custo extremamente elevado, do ponto de vista de recursos computacionais, pelo que, na prática, estas não são renderizadas na medida em que são solicitadas. Em vez disso, *tiles* para todos os níveis de zoom existentes são pré-renderizadas semanalmente, e armazenadas no servidor, para que possam ser servidas rapidamente, como imagens estáticas [22].

Conforme abordado por Neis [23], o projeto OSM tem vindo a ganhar popularidade, sobretudo a nível empresarial, após mudanças no modelo de licenciamento do Google Maps, no início de 2012, que significavam um potencial aumento de custos. Por outro lado, as organizações empresariais que disponibilizam plataformas equiparadas ao projeto OSM, reconheceram o potencial da informação gerada por utilizadores, pelo que, toda a informação contida nestas plataformas, incluindo dados submetidos por voluntários, são propriedade da respetiva plataforma, e não são disponibilizados ao público gratuitamente.

Visto que os dados disponibilizados pelo projeto OSM são inseridos por utilizadores registados, sem que estes tenham de ser peritos na aquisição de informação geográfica, é natural que surjam questões relativas à qualidade dos dados. Um estudo realizado por Haklay [24] pretende avaliar a qualidade destes dados, por comparação com *datasets* equiparados, provenientes de outras fontes, em termos de precisão e plenitude de informação. Um dos aspetos salientados, é a rapidez com que a informação contida no *dataset* OSM testado, foram recolhidos, tendo em conta o número de participantes. Relativamente à qualidade dos dados, o estudo demonstrou que esta pode atingir valores bastante satisfatórios, no entanto, isto nem sempre acontece, visto que a qualidade dos dados depende, em grande medida, do cuidado e atenção do utilizador, aquando da recolha dos dados.

2.7.1 OSM Tile Server

Conforme referido anteriormente, a renderização de *tiles* é um processo dispendioso, em termos de recursos, para os servidores. Além do mais, os servidores OSM dependem inteiramente de doações, resultando em capacidades limitadas. O uso extensivo de *tiles* OSM afeta negativamente os trabalhos de edição de mapa por parte dos utilizadores voluntários do projeto OSM, representando também um abuso relativo às doações individuais e patrocínios que providenciam *hardware* e largura de banda. Assim sendo, existe uma política restritiva de utilização de *tiles* particularmente em relação ao descarregamento das mesmas, para posterior utilização em modo *offline*. Genericamente, são proibidos quaisquer pedidos de descarregamento de *tiles* correspondentes aos níveis de *zoom* 17, ou maiores, a menos que seja obtida

permissão para tal, da parte de um administrador do sistema, uma vez que estes níveis de *zoom* não existem em *cache*, tendo de ser especificamente renderizados para cada pedido. A violação destas condições resulta no bloqueio de acesso ao serviço de *tiles* [25].

O *OSM Tile Server* consiste num conjunto de programas e bibliotecas, as quais em conjunto implementam um servidor de *tiles* [26]. Na sua implementação padrão, a qual é utilizada no principal servidor de *tiles* *OpenStreetMAp.org*, este sistema é composto por cinco componentes distintos, os quais encontram-se representados no diagrama presente na figura 2.2, a qual corresponde a uma secção do diagrama representativo da arquitetura dos componentes de todo o sistema do projeto OSM [27].

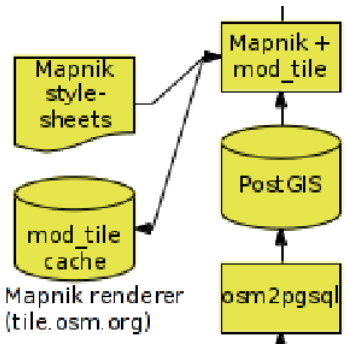

Figura 2.2: Componentes do *OSM Tile Server*

Uma base de dados *PostgreSQL*, ativada espacialmente pela extensão *PostGIS* é utilizada para assegurar o armazenamento da informação geográfica OSM, destinada à renderização da cartografia base. A importação desta informação para a base de dados com recurso à ferramenta *osm2pgsql* [28]. Esta, corresponde a um programa baseado em linha de comandos, o qual permite converter informação geográfica OSM para bases de dados *PostgreSQL*, espacialmente ativadas pela extensão *PostGIS* [29], sendo que o esquema de dados utilizado corresponde ao padrão de importação de dados OSM, para que possam ser utilizados por software de renderização [28].

Mapnik é um conjunto de ferramentas, de código aberto, destinado à renderização de mapas. É capaz de receber como input, informação geográfica codificada em diversos formatos, tais como *shapefile*, PostGIS, TIFF, ficheiros *osm*, CSV, e quaisquer formatos suportados pela biblioteca GDAL. O respetivo output consiste em imagens de mapa, as quais podem vir em diversos formatos, tais como PNG, JPEG, SVG ou PDF. A utilização do *Mapnik* por parte do OSM envolve a renderização de *tiles* de dimensão 256 x 256 pixeis. A renderização é baseada em regras de estilo definidas num formato XML específico do *Mapnik*.Existem conjuntos de regras de estilo *open source*, entre os quais o estilo *Mapnik* padrão, definido para a renderização de informação geográfica OSM. Os estilos podem também ser personalizados com recurso a algumas ferramentas externas, tais como *Cascadenik*, *Spreadnik* ou *TileMill*, as quais disponibilizam linguagens de estilo mais compactas e de interpretação mais acessível que o XML do *Mapnik* [30].

O serviço de *tiles* de cartografia base ao cliente é assegurado pelo *mod_tile*. Este sistema providencia funcionalidades de *caching*, e renderização dinâmica. Devido à renderização dinâmica, apenas existe necessidade de armazenar um número reduzido de *tiles* em disco, o que permite uma redução de recursos de armazenamento. Ao mesmo tempo, a sua estratégia de *caching* permite um serviço de alta performance, que pode suportar milhares de pedidos por segundo. Este sistema é composto por duas partes distintas. O *mod_tile* corresponde a um módulo *Apache* personalizado, que é responsável pelo serviço de *tiles*, e pedidos de renderização de novos *tiles*, que não existam em *cache*, ou que tenham sofrido alterações. O *renderd*

corresponde a um *backend* de renderização, que recebe pedidos do *mod_ tile*, e renderiza os mesmos, no sistema de ficheiros. É também responsável pela definição da fila de prioridade dos pedidos, de forma a impedir a sobrecarga do do servidor, face à situação de múltiplos pedidos efetuados simultaneamente [31].

2.8 Trabalho Relacionado

Existem trabalhos relacionados com o planeamento de circuitos pedestres e cicláveis, que se relacionam com o trabalho proposto neste documento.

Um destes trabalhos diz respeito ao desenvolvimento de uma aplicação destinada ao planeamento de percursos cicláveis [32], que utiliza *Web Services* e tecnologias assíncronas, e permite a escolha dos trajetos segundo diversos critérios. A aplicação permite critérios de seleção compensatórios e não-compensatórios.

A abordagem compensatória baseia-se em assumir que a alta performance de uma alternativa, atingida de acordo com um ou mais critérios pode compensar performances mais reduzidas para a mesma, segundo outros critérios. Pelo contrário, a abordagem não-compensatória define que a baixa performance de uma alternativa, segundo um critério, não pode ser compensada pelo resultado da aplicação de outros critérios [33]. Por outras palavras, pela abordagem compensatória, a pesquisa considerará alternativas menos satisfatórias aos critérios do utilizador, mas que poderão ter interesse para o mesmo. Um exemplo da aplicação desta abordagem seria no caso em que um utilizador, não tendo bem a certeza do que pretende pesquisar, adiciona outros parâmetros à pesquisa, podendo assim obter resultados que, não correspondendo às melhores opções retornadas pelo parâmetro inicial, poderão eventualmente enquadrar-se com o seu interesse. No caso da pesquisa pela abordagem não compensatória, são adicionados parâmetros a excluir dos resultados. Um exemplo desta abordagem seria o caso em que um utilizador sabe exatamente o que quer, ou mais concretamente, o que não quer, marcando à partida determinados parâmetros a excluir da pesquisa, para evitar o retorno de resultados que não se enquadrem nos seus interesses.

Para a determinação de percursos ótimos, é utilizada uma alternativa ao algoritmo *Shortest Path Analysis* (SPA), que recorre à utilização de topologia, tornando-se mais eficiente, em termos de processamento [32]. Esta abordagem determina, sequencialmente, as impedâncias dos arcos adjacentes a um nó de partida, e, tendo determinado o arco de menor peso, salta para o nó seguinte, adjacente ao referido arco, guardando-o, repetindo-se o mesmo processo, até que se chegue ao destino. No final obtém-se o percurso otimizado, segundo os parâmetros escolhidos [32].

A utilização de tecnologia de comunicação assíncrona permite enviar dados ao cliente em qualquer altura, não estando dependente dos inputs do utilizador, o que reduz significativamente o atraso na entrega de dados ao cliente [32].

Outro trabalho relacionado a referir, propõe o protótipo de uma aplicação de apoio à navegação, de carácter colaborativo, para pedestres, baseada em mapas gerados pelos utilizadores e classificação de percursos [34]. Esta aplicação permite aos utilizadores classificar segmentos de vias, de acordo com a acessibilidade que estas providenciam, informação essa que determinará a impedância dos segmentos dos percursos, para a determinação do melhor caminho. Isto significa que o algoritmo de pesquisa evita a escolha de caminhos com classificação negativa. A fiabilidade deste sistema de classificação depende da boa utilização do mesmo, estando sujeito a utilização maliciosa, nomeadamente, classificações falsas, que levantam problemas óbvios na determinação dos trajetos. Uma alternativa apontada para lidar com esta situação, é utilizar sempre o *feedback* mais recente na determinação dos trajetos, permitindo assim, minimizar o problema.

Um trabalho descrito em [35], utiliza velocidades para estradas, obtidas por GPS e sensores de trânsito, e histórico de percursos dos condutores, para planeamento de rotas. Uma questão importante abordada por este trabalho, é a dificuldade em coincidir as leituras de GPS com a realidade, visto que os recetores convencionais tem uma margem de erro na ordem dos 10 metros. É proposta uma abordagem para lidar com este problema, com recurso a um algoritmo que seleciona a melhor alternativa para para efetuar o *snap* de uma observação GPS com uma estrada. Esta questão considera-se relevante para a aplicação proposta no presente artigo, mas note-se que esta tem por objetivo lidar com percursos fora de estrada, existindo probabilidades reduzidas de estes se encontrarem cartografados, não existindo assim, uma base para efetuar o *snapping*. Assim sendo, poderão ser encontradas outras alternativas para lidar com a questão da precisão dos recetores GPS, podendo mesmo ser considerada a hipótese da margem de erro de 10 metros ser, de facto, aceitável.

A utilização das velocidades médias nas estradas, proposta em [35], para a determinação de trajetos mais rápidos, baseia-se em períodos horários distintos, nomeadamente horas-de-ponta e horas-mortas, visto que são fatores que influenciam a determinação dos trajetos mais rápidos. Ou seja, o mesmo trajeto, pode ser o mais rápido num determinado período, e não o ser noutro. Para a aplicação proposta neste artigo, as velocidades médias nos trajetos são relevantes, embora não seja de grande interesse que estas sejam determinadas por períodos horários, pois aos trajetos fora de estrada, não faz sentido aplicar os conceitos de hora-de-ponta, da mesma forma que nas estradas com grande movimentação.

Uma implementação de um procedimento automático, destinada a gerir rotas de veículos pesados, descrita em [36], utiliza, para a determinação das rotas, utiliza duas rotinas que incluem o algoritmo de Dijkstra (procura do caminho mais curto), em conjunto com a verificação de parâmetros relacionados com as características do veículo em questão, comparando-os com os valores permitidos para a travessia de pontes, ou passagens sob viadutos. Esta rotina repete-se, até que se encontre o caminho mais curto possível, ou o caminho que permita a maior capacidade de carga, de acordo com os parâmetros do veículo. As dimensões dos veículos são relevantes para a aplicação apresentadas neste artigo, particularmente para satisfazer as necessidades das corporações de bombeiros, permitindo a estes verificar, caso exista, a possibilidade de utilizar veículos de maior dimensão. A intenção passa por aplicar uma abordagem semelhante à descrita em [36], definindo-se classes de veículos, e utilizando este parâmetro para encontrar percursos.

Capítulo 3

Solução de Partilha, Enriquecimento e Pesquisa de Percursos

A aplicação aqui proposta foi desenvolvida tendo em vista a reutilização de tecnologias e soluções já existentes, combinadas de forma a permitir a partilha e pesquisa de percursos, permitindo ainda acrescentar outras informações úteis, de forma a enriquecer as capacidades de pesquisa.

Pretende-se que esta aplicação permita armazenar e servir informação geográfica referente a percursos, acima de tudo, em que estes se possam relacionar com POIs, enriquecendo assim a informação relativa aos percursos, sem a necessidade de alterar a estrutura de dados dos mesmos. Desta forma, os percursos, carregados para a aplicação num determinado formato standard, podem posteriormente alvo de enriquecimento de informação, através de marcação de POIs associados aos mesmos, a qual será utilizada sob forma de diversos critérios de pesquisa de percursos, retornando os resultados com a mesma estrutura de dados em que os mesmos foram carregados, preservando assim, acima de tudo, o standard utilizado.

O presente capítulo descreve, ao longo das primeiras três secções, os tipos de informação envolvidos nesta aplicação, abordando também a utilidade que representam, a respetiva finalidade, passando também pelo armazenamento em base de dados. A quarta secção descreve, ao longo das respetivas subsecções, as funcionalidades implementadas que permitem inserir ou criar, e disponibilizar a dita informação, conforme se encontram de momento. A figura 3.1 representa os referidos tipos de informação, assim como de que forma se relacionam.

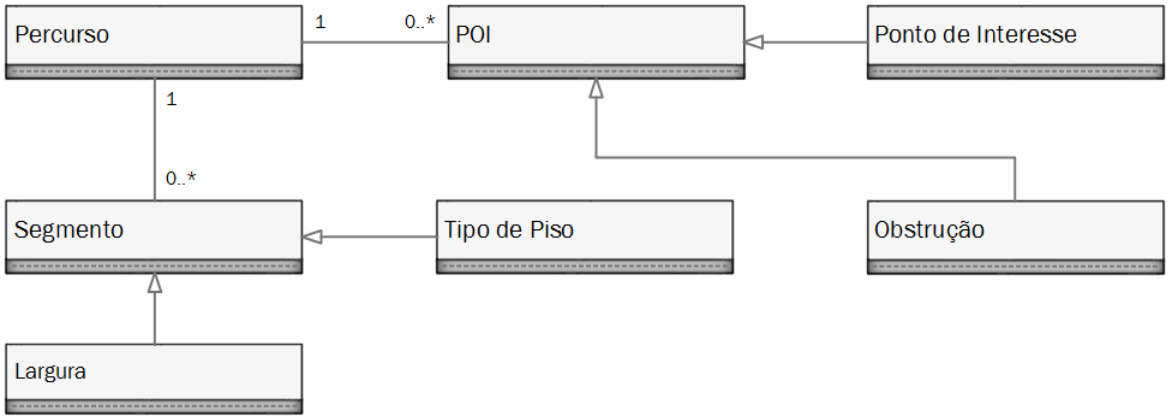

Figura 3.1: Tipos de informação envolvidos na aplicação, e respetivo relacionamento entre os mesmos.

3.1 Percursos

Os percursos correspondem ao principal tipo de informação que a aplicação desenvolvida tem como motivação servir. São inseridos por utilizadores, e, após operações de pesquisa, são retornados aos mesmos, no mesmo formato em que foram inseridos no sistema. Assim sendo, o que se pretende com o retorno de uma operação de pesquisa, é um conjunto de percursos, de acordo com os parâmetros de pesquisa escolhidos, e caso o utilizador introduza múltiplos parâmetros, que este conjunto fique ordenado, relativamente às suas preferências.

Para além da informação pontual, descrita na secção anterior, e que serve como parâmetros de pesquisa, considera-se também a utilidade de incluir dados adicionais correspondentes a segmentos de percurso, visto que, certos critérios de pesquisa, como o tipo de piso e a largura do trilho, correspondem algo que existe numa determinada extensão do percurso, e não em apenas um ponto.

3.2 Segmentos de Percurso

Correspondem, como o próprio nome indica, a porções de um determinado percurso, com uma determinada característica. Permitem guardar informação de carácter não pontual, como por exemplo, inclinação média, velocidade média, tipo de piso, largura do percurso, entre muitas outras hipóteses, numa determinada secção do percurso, permitindo assim, uma gama ainda mais vasta de opções relevantes de escolha, na pesquisa de percursos.

São representados espacialmente por linhas, assim como os percursos, o que permite efetuar pesquisas em relação à extensão apresentada por uma determinada característica nos percursos, para além do número de ocorrências dessa característica apresenta no mesmo percurso. Isto significa que, para o caso de se pretender pesquisar percursos com base num determinado tipo de piso, o resultado dessa pesquisa pode ser ordenado com base no somatório das distâncias dos segmentos do mesmo tipo que existem por percurso retornado, em vez de se limitar a ordenação de resultados de acordo com o número de segmentos do mesmo tipo existentes em cada percurso.

As seguintes categorias de segmentos foram consideradas, para efeitos de implementação de funcionalidades de pesquisa, e testes aos resultados:

- **Largura** - Segmentos estreitos, de largura inferior a 2.5m.
- **Tipo de Piso**
  - Terra
  - Lama ou Água
  - Gravilha
  - Rocha

Em relação à largura, consideram-se duas classes, partindo do princípio que a largura do percurso representa, ou não, dificuldades à passagem dos veículos. Largura inferior a 2.5m, para segmentos de percurso estreitos, e superior a 2.5m, para segmentos largos, no entanto, encontram-se definidos apenas os segmentos estreitos. Seria, de certa forma, redundante, e desnecessário, guardar segmentos para ambas as classes de largura do percurso, pois para este caso específico, subentende-se que todas as secções de um percurso que não correspondam a segmentos estreitos, são, logicamente, segmentos largos. A pesquisa de percursos de acordo com o parâmetro largura pode perfeitamente ser realizada, unicamente com base nos segmentos estreitos. Caso o utilizador tenha preferência por percursos largos, a pesquisa pode

ser efetuada omitindo-se percursos com segmentos estreitos, ou ordenando os resultados de forma ascendente, consoante o número de segmentos por percurso, ou a distância que estes representam, aparecendo assim, em primeiro lugar, os percursos que melhor se enquadram nas preferências do utilizador. Caso contrário, podem-se omitir percursos que não contenham segmentos estreitos, ou ordenam-se os resultados de forma descendente, de acordo com o número de segmentos por percurso, ou distância que estes representam.

Dados sobre a largura do percurso, ou o tipo de piso, embora não sejam cruciais para os praticantes experientes de TTT, proporcionam a estes, uma razoável impressão das condições que poderão encontrar no terreno. São, no entanto relevantes para determinar que tipos de veículo tem capacidade para circular em determinados percursos, o que se entende ser do interesse das corporações de bombeiros, e Proteção Civil, para o planeamento do combate a incêndios florestais.

3.3 Pontos de Interesse

Os Pontos de Interesse representam uma alternativa viável para armazenar e disponibilizar informação sobre artefactos de carácter pontual, relacionados com os percursos. De referir como exemplo para os mesmos, obstruções no percurso, ou locais de relativa importância para quem utilize o percurso.

Para a aplicação aqui descrita, pode-se considerar uma gama relativamente vasta de categorias de Pontos de Interesse, no entanto, tendo em vista a implementação de funcionalidades de pesquisa, e realização de testes às mesmas, foram consideradas, numa primeira fase, as seguintes categorias:

- **Obstruções**
  - **Quedas de árvores**
  - **Derrocadas**
- **Pontos de interesse**
  - **Curvas apertadas**
  - **Zonas de inversão de marcha**
- **Segmentos do percurso, relativos a um tipo de piso**
  - **Terra**
  - **Gravilha**
  - **Lama ou água**
  - **Rocha**
- **Segmentos de percurso, relativos à sua largura**

Como o próprio termo indica, as obstruções tem como finalidade fornecer informação sobre o estado de transitabilidade do percurso. Independentemente do grau de dificuldade do mesmo, um utilizador pode, caso existam obstruções num percurso, ver que tipo de obstrução ocorreu, a sua localização geográfica, assim como a data em que esta foi inserida na base de dados. São assim fornecidas informações atualizadas sobre o estado de transitabilidade dos percursos, assim como informações sobre como lidar com as obstruções, tendo em conta que haja interesse em remover as mesmas do terreno.

Informações sobre curvas apertadas e zonas de inversão de marcha são relevantes, particularmente para corporações de bombeiros e Proteção Civil. Visto que estes poderão ter interesse em recorrer a percursos TT, durante o combate a incêndios florestais, pontos de interesse deste tipo ajudam a determinar a viabilidade dos percursos, para veículos de grande dimensão.

Os segmentos de percurso correspondem a um caso particular, abordado na secção anterior. É de salientar, no entanto, que cada segmento é determinado por dois pontos, correspondendo estes às extremidades do segmento. Assim sendo, a aplicação aqui descrita recorre à solução dos Pontos de Interesse, para definir o início e o fim de um determinado segmento de percurso, associando aos mesmos a respetiva informação sobre o tipo de piso existente nesse segmento de percurso, ou à respetiva largura.

3.4 Funcionalidades Implementadas

A presente secção providencia uma descrição completa de como as funcionalidades se encontram implementadas, neste momento.

3.4.1 Inserção de Percursos

Esta funcionalidade está implementada de forma a extrair a informação relativa aos percursos, contida em ficheiros GPX, inserindo-a, de seguida na base de dados. Este processo implica o recurso a uma base de dados temporária e uma inserção de dados feita de forma sequencial, existindo assim a necessidade de restringir o acesso à base de dados a um utilizador de cada vez, para evitar o cruzamento de dados de múltiplas sessões simultâneas, garantindo assim a integridade destes. Segue-se uma descrição detalhada de todo o processo, desde a ação do utilizador, passando pela extração de dados, por parte do servidor, terminando na inserção dos mesmos, na base de dados.

O carregamento de ficheiros GPX é assegurado pela extensão *JQuery File Upload*, tendo sido implementado com base em exemplos disponíveis em *hayageek.com* [37]. Esta extensão permite, para além de diversas outras opções, restringir o envio de ficheiros ao servidor de acordo com o formato dos mesmos. Assim sendo, basta definir o formato GPX como único formato aceite, ficando assim esta questão fechada do lado do cliente, não havendo necessidade de verificar o formato dos ficheiros pelo servidor.

O tratamento e inserção da informação recebida é feita em duas etapas, envolvendo o recurso à ferramenta *ogr2ogr*, existente na biblioteca GDAL, assim como uma base de dados temporária.

No servidor, os ficheiros são recebidos por um *script* denominado *recieve_files.php*, o qual corresponde à primeira etapa de inserção. A figura 3.2 representa a sequência de eventos associada à execução deste *script*.

A verificação dos ficheiros recebidos trata de dois casos distintos que poderão ocorrer na invocação deste *script*. Ou foi enviado apenas um ficheiro, ou um *array* contendo múltiplos ficheiros. Na prática, é executada a mesma rotina para cada ambos os casos, em que a diferença está no facto da referida rotina ser executada dentro de um ciclo, o qual itera por todos os elementos do array de ficheiros, isto para o caso de ocorrer esta situação.

O *lock file* aqui referido tem por objetivo impedir que ocorram cruzamentos de dados provenientes de outras sessões. Pretende-se que, quando múltiplos utilizadores tentem submeter percursos simultaneamente, apenas uma sessão de cada vez possa aceder à base de dados, ficando as restantes em espera.

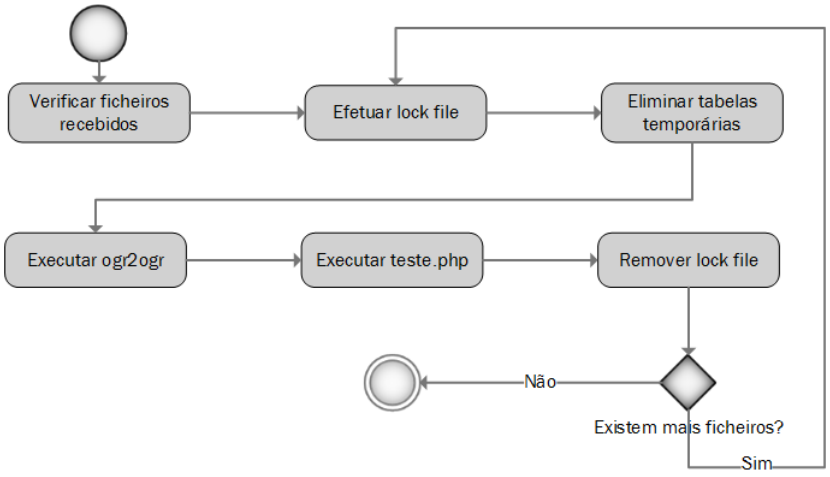

Figura 3.2: Diagrama de atividades correspondente ao *script recieve_files.php*

O passo seguinte, corresponde à inserção dos percursos existentes em cada ficheiro na base de dados temporária. Esta deverá estar vazia, antes da inserção, caso contrário poderão ser inseridos dados de percursos que, na realidade não correspondam àquilo que o utilizador pretende inserir. Para garantir que tal situação não ocorra, qualquer informação que possa eventualmente existir nesta base de dados é apagada, antes de ser efetuada qualquer inserção.

A inserção de dados é feita com recurso à ferramenta *ogr2ogr* incluindo os respetivos parâmetros necessários, destacando-se a ligação à base de dados temporária, assim como o ficheiro a introduzir.

Finalmente, é invocado um segundo *script*, neste caso denominado *teste.php*, o qual se encarrega da segunda etapa de inserção.

Caso o utilizador tenha carregado múltiplos ficheiros, este processo repete-se até que todos os ficheiros sejam inseridos.

A segunda fase de inserção, em que os percursos são transferidos da base de dados temporária para a final implica, conforme fora referido anteriormente, um processo de inserção sequencial. Como resultado da execução da ferramenta *ogr2ogr* resultam dois conjuntos de dados distintos, no entanto, relacionados. *Tracks* e *track_points*, em que múltiplos elementos do segundo conjunto encontram-se relacionados com um elemento do primeiro. Ao inserir novos percursos na base de dados final, é necessário preservar a sequência de valores de chave primária nas tabelas da base de dados final, assim como atribuir valores corretos para a chave estrangeira dos *track_points*, garantindo assim a correta relação entre estes e os respetivos *tracks*. A figura 3.3 mostra o diagrama de atividades correspondente ao *script teste.php*, o qual se encarrega destas operações.

A sequência de inserção dos percursos na base de dados final consiste, sumariamente, em inserir um *track* distinto, seguido dos respetivos *track_points* correspondentes, estabelecendo, por fim, a relação entre estes.

Desta forma, são obtidos em primeiro lugar todos os elementos presentes na tabela *tracks* da base de dados temporária, para que estes possam ser iterados, realizando as restantes operações em cada iteração.

Por cada *track* obtido, isola-se o valor da correspondente chave primária, o qual será posteriormente necessário, para que sejam obtidos os *track_points* correspondentes. Este valor vem diretamente do elemento correspondente ao valor da chave primária do *track* que está a ser iterado.

Neste momento, o *track* já pode ser inserido na base de dados final. É necessário conhecer

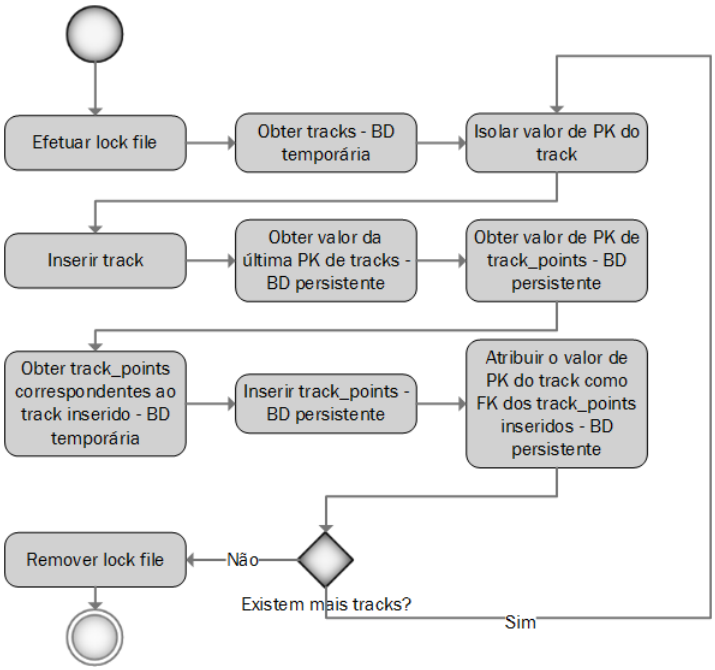

Figura 3.3: Diagrama de atividades correspondente ao *script teste.php*

o valor de chave primária atribuído ao *track* inserido na base de dados final, para que posteriormente seja possível relacionar os *track_points* relativos a este *track*, com o mesmo, após a inserção destes. A solução implementada para obter este valor recorre à combinação das funções *currval* e *pg_get_serial_sequence*, ambas implementadas no SGBD, o que permite retornar o valor mais recente de uma sequência, sendo o caso da chave primária.

Antes de se proceder à inserção dos *track_points* na base de dados final, é também necessário conhecer o valor da chave primária do último *track_point* lá existente, visto que, após a inserção tem de ser estabelecida a relação entre estes e o *track* correspondente, o que implica modificar os valores da chave estrangeira para um determinado conjunto de *track_points*. Assim sendo, se se conhecer o valor da chave primária do último *track_point* inserido previamente, antes de serem inseridos novos, após a referida inserção basta atualizar os valores da chave estrangeira de todos os *track_points* com valor de chave primária superior ao valor obtido previamente, os quais corresponderão ao último *track* inserido.

Este processo é repetido para cada *track* obtido da base de dados temporária, finalizando-se quando todos os *tracks* e respetivos *track_points* se encontrem inseridos na base de dados, terminando também todo o processo de inserção de percursos descrito na presente subsecção.

3.4.2 Inserção de Obstruções e Pontos de Interesse

Ao contrário dos percursos, estes dados são inseridos por interação do utilizador com a própria aplicação, em vez de serem carregados a partir de ficheiros.

A interface cliente disponibiliza um formulário no qual duas opções distintas podem ser utilizadas para a obtenção das coordenadas do ponto a inserir, a localização do dispositivo e a obtenção de coordenadas por clique no mapa. Independentemente da opção utilizada, o utilizador deverá ter um percurso pré-selecionado, caso contrário, não será possível submeter a obstrução ou ponto de interesse, visto que este tipo de informação pontual está sempre associada a um percurso.

O recurso à localização do dispositivo para obtenção das coordenadas do ponto, tem em

vista a utilização desta aplicação em dispositivos móveis, no próprio terreno. O botão *Minha localização* recorre a este método para obter as coordenadas do ponto, colocando os respetivos valores nos campos *Latitude* e *Longitude*. Esta operação recorre uma extensão da biblioteca *JQuery*, denominada *JQuery-Geolocation*.

A opção de obter as coordenadas do ponto através de clique no mapa foi desenvolvida sobretudo para facilitar a realização de testes á aplicação, visto que, desta forma, não há qualquer necessidade de deslocações ao terreno. Em vez disso, o utilizador ativa a *checkbox* denominada *Obter localização do mapa*, efetuando, de seguida, um clique nas proximidades do percurso selecionado. Esta operação coloca também o valor das coordenadas nos campos *Latitude* e *Longitude*, no entanto, recorre ao evento *click* do objeto mapa, definido pela biblioteca *Leaflet*.

Relativamente à secção *Informações Base*, a qual corresponde aos dados alfanuméricos relativos ao ponto, encontram-se, de momento, definidos os campos *Nome* e *Categoria*. Estes campos, conjuntamente com a localização, correspondem aos parâmetros obrigatórios para que seja introduzido um POI com sucesso no sistema, através da API do *backend* POI-DP. O nome a atribuir ao ponto fica à descrição do utilizador, enquanto que a categoria do mesmo, pode ser selecionada de uma lista de opções pré-definidas.

O processo de inserção da obstrução ou ponto de interesse começa ainda no cliente da aplicação, visto que a informação a enviar tem de ser codificada num objeto JSON, o qual possa ser interpretado corretamente pelo *backend* POI-DP. Assim sendo, após o formulário se encontrar preenchido, o botão *Submeter* dá início ao processo descrito no diagrama de atividades representado na figura 3.4.

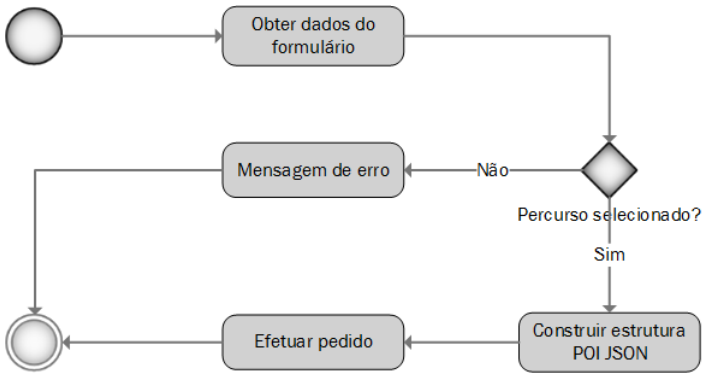

Figura 3.4: Diagrama de atividades correspondente ao processo de codificação do ponto em objeto JSON, e respetiva submissão

Testa-se a existência de um percurso selecionado, verificando o valor de uma variável global, a qual deverá conter o *id* de um percurso existente no cliente. Caso a variável contenha uma *string* vazia, significa que nenhum percurso foi selecionado, retornando assim uma mensagem a informar acerca da situação, terminando-se assim o processo de inserção. Caso esta variável contenha um valor numérico, existe um percurso selecionado. Assim sendo, procede-se à construção do objeto JSON, o qual irá incluir, para além do *timestamp*, o referido *id* do percurso. Posto isto, a função retorna o objeto JSON. Note-se que o referido objeto é construido tendo em conta o esquema JSON para POIs presente na especificação da API do *backend* POI-DP [20], ao qual foi incluído um componente extra, denominado *poi_extra_data*, o qual vai conter o *id* do percurso correspondente, e a categoria. Este componente será abordado em maior detalhe quando forem abordados os *scripts* do servidor.

Este objeto JSON, o qual representa a obstrução ou ponto de interesse, é enviado para o servidor através da invocação de uma função, a qual recebe o referido objeto como parâmetro,

baseada num exemplo retirado do guia de utilização do *backend* POI-DP [38], destinado a adicionar POIs, no entanto, foi feita uma modificação. Em vez de se efetuar um pedido POST para o *script add_poi.php*, o qual corresponde à implementada no *backend* para inserir um novo POI na base de dados, este é direcionado para *my_add_poi.php*, o qual executa um conjunto de operações descrito no diagrama de atividades representado na figura 3.5.

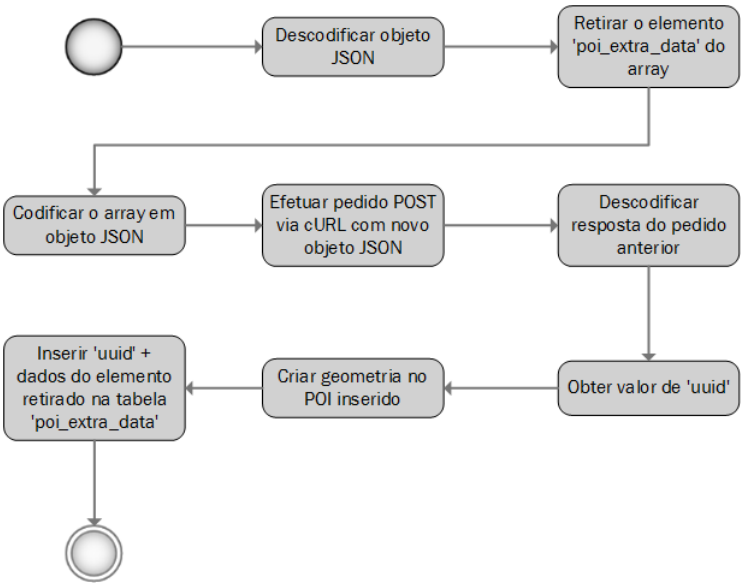

Figura 3.5: Diagrama de atividades correspondente ao *script my_add_poi.php*

Este *script*, fundamentalmente desempenha a tarefa de retirar a informação extra do objeto JSON recebido, enviando-o, de seguida, com a estrutura correta para o *script add_poi.php*, para que o POI seja introduzido na base de dados. Posto isto, insere a restante informação na tabela extra, denominada *poi_extra_data*.

Uma vez recebido e descodificado o objeto JSON, procura-se a chave do elemento a retirar, com recurso ás funções *array_search* e *array_keys*, para, de seguida retirar o referido elemento com recurso à função *array_splice*. Desta operação resultam dois *arrays*, um dos quais correspondendo aos dados a inserir via *script add_poi.php*, outro correspondendo aos dados a inserir posteriormente na tabela extra.

De seguida, o primeiro dos *arrays* referidos anteriormente é codificado em objeto JSON, e enviado através de um pedido POST, realizado com recurso à biblioteca cURL, para *add_poi.php*, o qual retorna outro objeto JSON, contendo o valor do campo *uuid*, que corresponde à chave primária do POI inserido, e o respetivo *timestamp* da inserção. Neste caso, o *uuid* do POI é importante, para estabelecer a relação entre os dados extra, que ficam armazenados numa tabela em separado, e o respetivo POI.

Antes de se proceder à inserção dos dados extra, existe uma operação importante a ser realizada. Note-se que as coordenadas do ponto são inseridas como Latitude e Longitude, ou seja, coordenadas geográficas. É fundamental que os POIs existentes na base de dados do *backend* tenham também coordenadas geométricas, uma vez que a grande maioria das funções espaciais da extensão PostGIS requer a geometria dos objetos espaciais para produzir resultados. Embora a tabela *fw_core* presente na base de dados do *backend* contenha uma coluna destinada às geometrias dos pontos, esta não é preenchida pelos *scripts* de inserção do *backend*. Assim sendo, encontra-se implementada uma *query* que, após a inserção de um POI, atualiza o campo *geometry* da tabela *fw_core*, com a respetiva geometria. Esta pode ser facilmente obtida a partir da geografia do POI, já presente na coluna *location*, através de um

cast, visto que, tanto geografias como geometrias, são tipos de dados definidos pela extensão PostGIS.

Resolvida a questão da geometria do POI, resta então, inserir os dados extra na tabela *poi_extra_data*. Esta tabela é responsável por estabelecer a relação entre os POIs e os respetivos percursos a que dizem respeito, desempenhando ainda um papel semelhante na criação de segmentos de percurso, que serão abordados na próxima subsecção. O valor de *uuid*, obtido anteriormente, conjuntamente com os valores de *id* do percurso e categoria do POI, retirados ao objeto JSON inicialmente recebido, são agora inseridos na tabela *poi_extra_data*, terminando-se assim o processo de inserção de POIs relativos a obstruções e pontos de interesse.

3.4.3 Criação de Segmentos de Percurso

A criação de segmentos de percurso reutiliza grande parte da implementação da inserção de obstruções e pontos de interesse. Apesar de os segmentos de percurso serem, na realidade linhas, o POI é utilizado para que um utilizador possa definir o início e o fim de um segmento num percurso. O servidor encarrega-se de gerar os segmentos, a partir dos POIs existentes no *backend* POI-DP relativos aos referidos segmentos correspondentes. Por fim, são inseridos em tabelas específicas para os mesmos, estabelecendo-se, por fim a relação entre estes e os percursos correspondentes.

Esta funcionalidade encontra-se implementada de forma a que um segmento seja sempre definido por dois pontos, correspondendo ao início e fim do mesmo. Assim sendo, realiza-se um conjunto de operações, descritas no diagrama de atividades representado na figura 3.6, as quais garantem o envio de dois POIs para um *script* desenvolvido especificamente para segmentos de percurso.

O processo inicia-se com uma listagem de percursos, a qual, na verdade funciona como uma verificação se existem ou não percursos carregados no cliente.

O processo de marcação dos pontos inicial e final do segmento a inserir encontra-se implementado de forma sequencial, em que o utilizador insere, em primeiro lugar, o ponto inicial do segmento, inserindo de seguida o ponto final, exclusivamente por esta ordem. Assim sendo, para a largura, como para o tipo de piso, os botões *Criar segmento* ativam um botão secundário, destinado à marcação do primeiro ponto. Uma vez marcado o ponto inicial, o referido botão secundário é substituído por outro semelhante, neste caso destinado à marcação do ponto final. Os referidos botões secundários representam uma sequência de operações, descrito no diagrama de atividades ilustrado na figura 3.7, que reutiliza funções implementadas para a inserção de POIs, terminando na criação de um *array* contendo dois objetos JSON com a mesma estrutura definida para POIs utilizada pelo *backend* POI-DP, e consequente submissão do mesmo para o servidor.

A criação de ambos os POIs utiliza um processo semelhante à funcionalidade de inserção de obstruções e pontos de interesse, descrita na subsecção anterior, em que as mesmas funções foram remodeladas, introduzindo-se ligeiras alterações. A obtenção de coordenadas do ponto efetua-se através de um evento *click* sobre o mapa, nas proximidades do percurso, onde o utilizador desejar marcar a extremidade do segmento. No entanto, em vez de se introduzir as coordenadas em campos de um formulário, estas passam como parâmetro, assim como os restantes dados constituintes do POI, na chamada de uma função destinada à construção do objeto JSON, correspondendo esta a uma versão modificada da função utilizada na funcionalidade descrita na subsecção anterior. Aparte das coordenadas, os restantes parâmetros inseridos correspondem ao *id* do percurso correspondente ao segmento, um parâmetro que determina se o POI corresponde ao ponto inicial ou final do segmento, e, por fim, o tipo de piso, em que este último apresenta um carácter opcional, ao contrário dos restantes. Isto significa

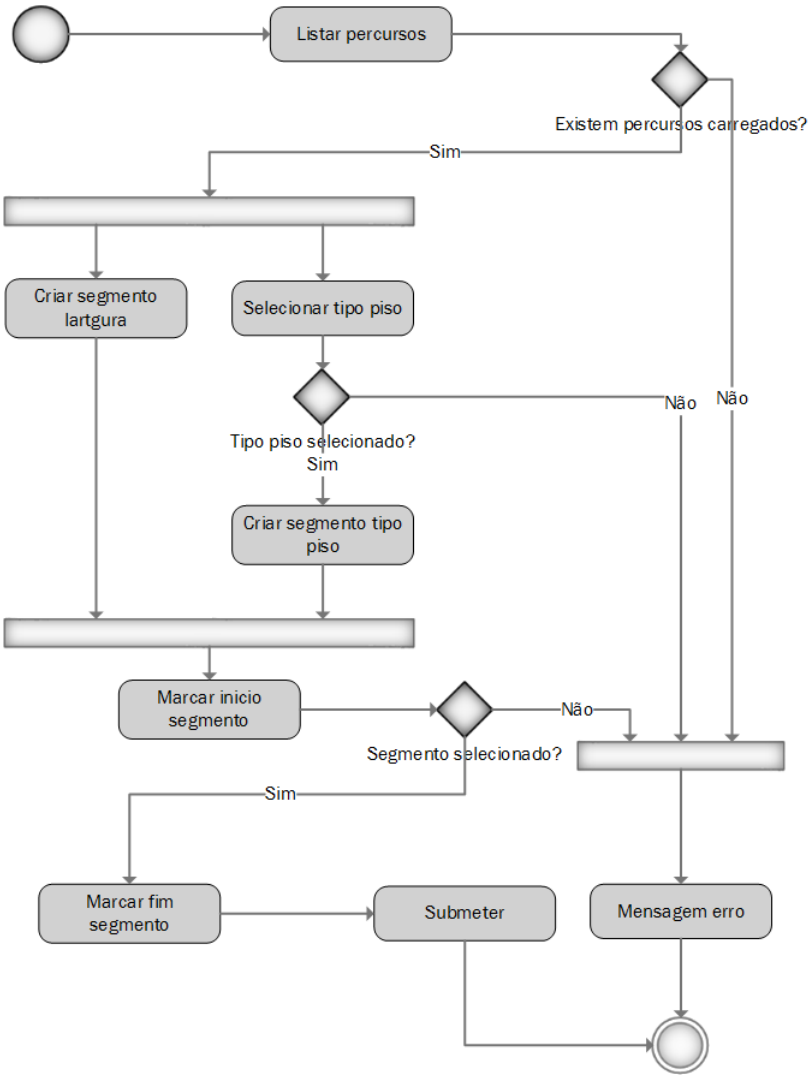

Figura 3.6: Diagrama de atividades correspondente às ações do utilizador para criação de um segmento de percurso

que a mesma função é invocada para a criação de ambos os tipos de segmento, sendo que, no caso de estar a ser criado um segmento de percurso relacionado com a largura do percurso, o a função é invocada sem o parâmetro correspondente ao tipo de piso. Caso contrário, o referido parâmetro, não só é incluído, como também contém a informação sobre o tipo de piso específico a que o segmento diz respeito.

Uma vez criado cada um dos POIs correspondentes aos pontos inicial e final do segmento, este é adicionado a um *array* denominado *segments*, o qual se encontra vazio, aquando da marcação do ponto inicial. Após ser adicionado o ponto final do segmento ao referido *array*, procede-se ao envio do mesmo para o servidor. Tal é efetuado através da invocação de uma função semelhante à mesma utilizada para submeter uma obstrução ou ponto de interesse, em que a única diferença reside na *url* de destino, visto que encontra-se implementado um *script* para lidar especificamente com a criação de segmentos.

Após a submissão, o conteúdo do *array segments* é apagado, para que este possa ser reutilizado corretamente para criar novos segmentos. O cliente disponibiliza também uma mensagem, em forma de *alert*, com a resposta do *backend* POI-DP, informando o utilizador

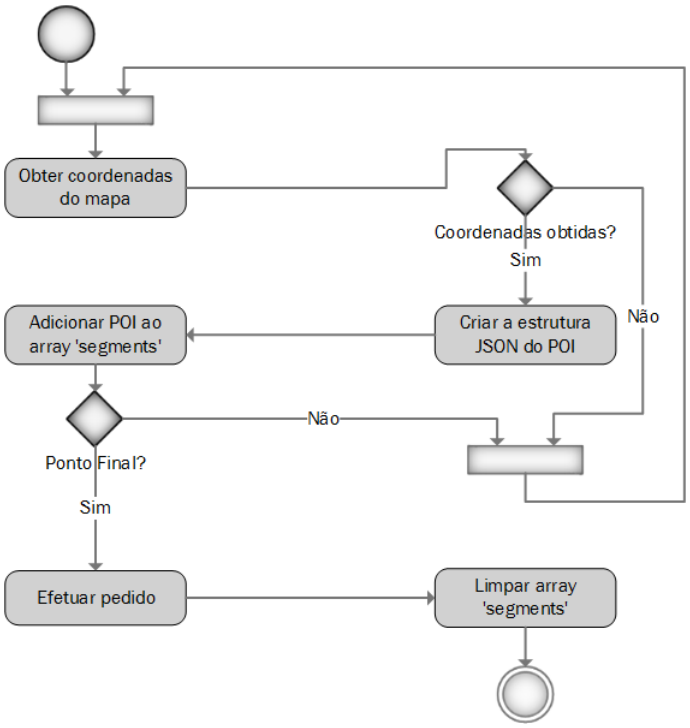

Figura 3.7: Diagrama de atividades correspondente à sequência de operações destinadas a criar um *array* de POIs, e efetuar a respetiva submissão

da inserção bem sucedida de POIs.

O *array* contendo os POIs correspondentes aos pontos inicial e final do segmento é enviado através de um pedido POST para um *script* denominado *my_ add_ poi_ segment.php*, o qual trata-se de uma modificação de *my_ add_ poi.php*, utilizado para a inserção dos POIs correspondentes às obstruções e pontos de interesse. Este *script* começa por implementar a mesma rotina de *my_ add_ poi.php*, por cada POI existente no *array*, acrescentando, após a inserção de ambos os POIs na base de dados do *backend* POI-DP, um novo conjunto de operações, destinadas à construção de um novo *array* de dados a submeter para outro *script*, o qual efetua o processo de criação do segmento. A figura 3.8 mostra um diagrama de atividades representativo do conjunto de operações efetuado pelo *script my_ add_ poi_ segment.php*.

Encontra-se implementada a mesma rotina de *my_ add_ poi.php*, no entanto, esta é repetida por cada elemento do array recebido. Note-se que, neste caso, é recebido um *array* contendo dois POIs, pelo que neste diagrama é utilizado o termo *sub-array*, o qual diz respeito a cada um dos POIs. Uma vez submetidos os POIs para o *backend* POI-DP, é construido um novo *array*, denominado *segment_ data*. Este, apresenta uma estrutura semelhante ao *array* de entrada, no entanto é composto pelo *uuid* de cada POI, conjuntamente com o respetivo componente *poi_ extra_ data*, sendo estas as informações necessárias para a criação do segmento de percurso. De seguida, o referido *array* é codificado em objeto JSON, e submetido para um novo *script*.

Mediante o tipo de segmento a criar, sendo este referente à largura do percurso, ou ao tipo de piso, o processo de criação do segmento apresenta ligeiras diferenças, pelo que se encontram implementados dois *scripts* distintos para efetuar a referida operação. Um para construir segmentos baseados na largura do percurso, e outro para segmentos baseados no tipo de piso, respetivamente, *segmentos_ largura.php* e *segmentos_ tpiso.php*. O *script* destino do pedido POST é definido pelo conteúdo dos campos *tipo_ piso* ou *largura_ trilho*, existentes

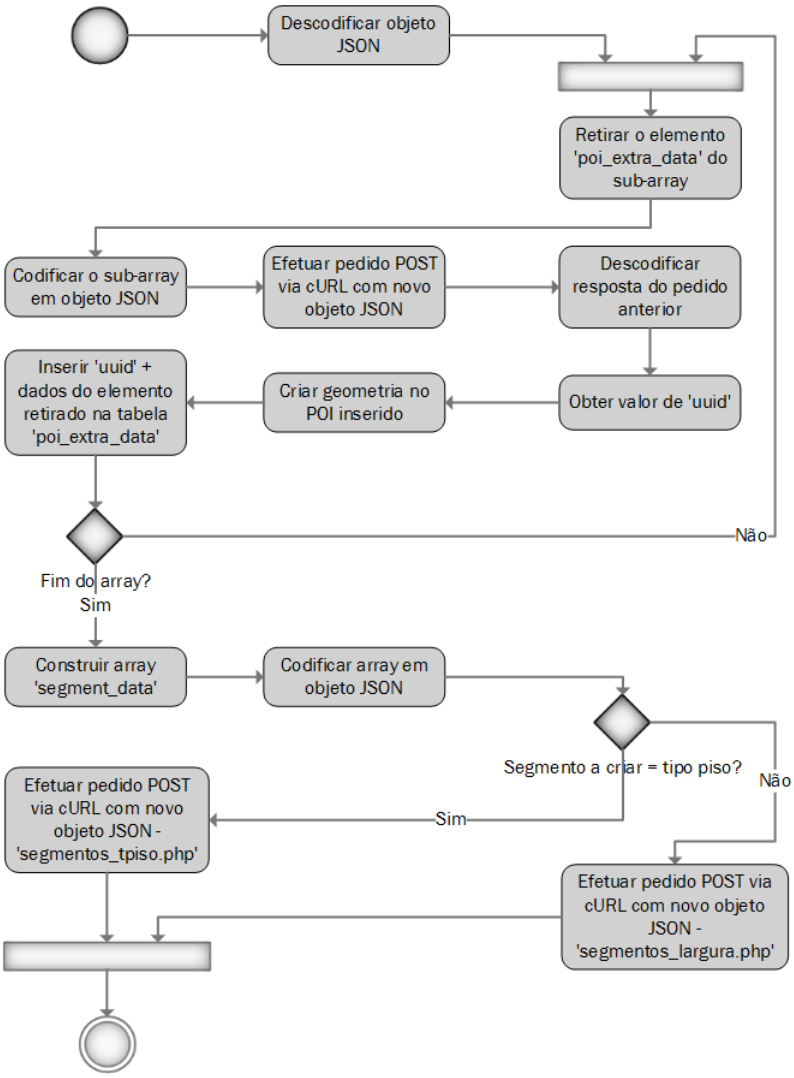

Figura 3.8: Diagrama de atividades correspondente à sequência de operações do *script my_ add_ poi_ segment.php*

em qualquer dos POIs. Um destes campos deverá estar sempre vazio, sendo que o restante deverá conter um valor. Caso o tipo de segmento a criar seja relacionado com o tipo de piso, então o campo *largura_ trilho* deverá estar vazio, invocando-se assim, o caminho para o *script segmentos_ tpiso.php* na url do pedido POST, por exemplo. Caso contrário, o campo *tipo_ piso* estará vazio, invocando-se consequentemente o caminho para *segmentos_ largura.php* na url.

Procede-se então à ultima etapa do processo de criação de um segmento de percurso, a qual implica a obtenção dos *track_ points* correspondentes ao mesmo, a partir dos quais será construida a geometria do segmento, que neste caso corresponde a uma linha. No final deste processo, a geometria criada para o segmento, em conjunto com a restante informação alfanumérica associada ao mesmo são inseridas numa de duas tabelas específicas, consoante o tipo de segmento, designadas por *segmentos_ largura* ou *segmentos_ tipo_ piso*. Uma vez que ambos os *scripts* referidos apresentam essencialmente a mesma rotina, sendo que o *script segmentos_ tpiso.php* apenas acrescenta uma operação extra, relacionada com o tipo de piso específico do segmento, será abordado com maior detalhe o *script segmentos_ largura*, focando-se, no fim, as operações extra existentes no *script segmentos_ tpiso.php*. O diagrama de atividades

representado na figura 3.9 descreve as operações executadas pelo *script segmentos_ largura*.

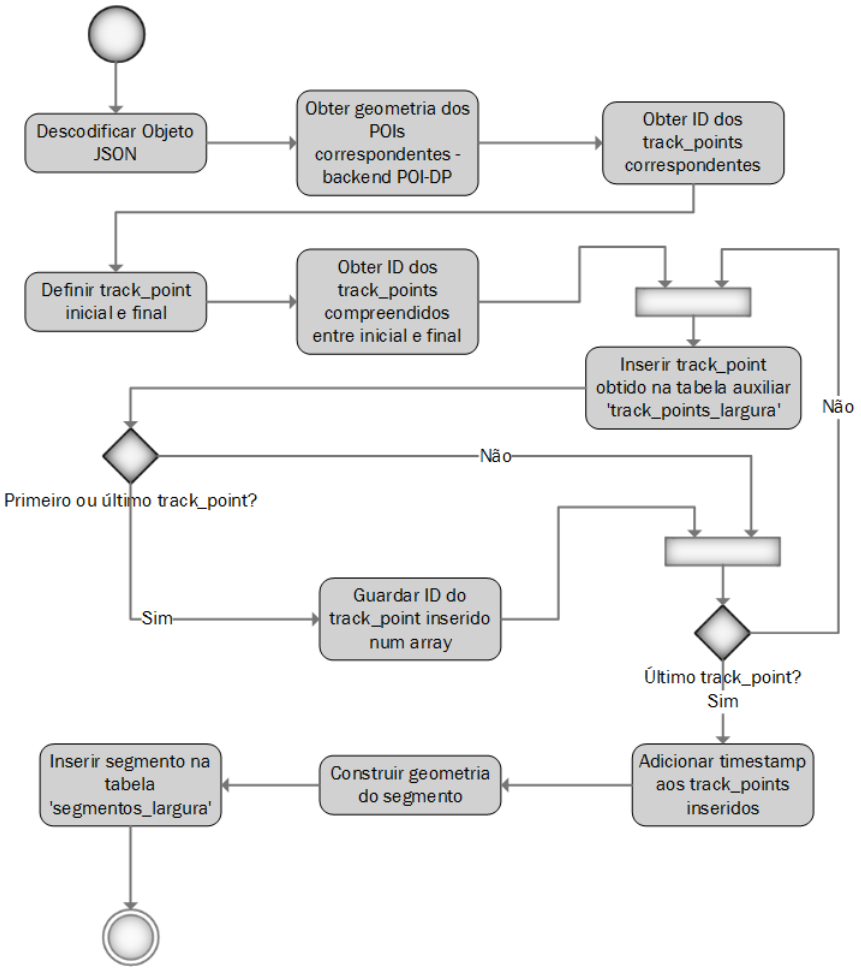

Figura 3.9: Diagrama de atividades correspondente à sequência de operações realizadas em *segmentos_ largura.php*

Após a descodificação do objeto JSON, procede-se à obtenção das geometrias correspondentes a ambos os POIs, a qual será necessária para determinar os *track_ points* correspondentes a estes no percurso. O objeto JSON contém os campos *uuid*, os quais correspondem à chave primária dos POIs, sendo estes valores utilizados para retornar as geometrias correspondentes, do *backend* POI-DP.

A geometria do segmento de percurso terá de ser criada a partir de um conjunto de pontos, os quais se encontram na tabela *track_ points* da base de dados dos percursos. A geometria dos POIs, obtida anteriormente, permite efetuar uma *query* de proximidade sobre as geometrias presentes nesta tabela, retornando, para cada POI, o respetivo track_point mais próximo, em termos espaciais. Obtém-se assim os valores de chave primária, para os *track_ points* inicial e final do segmento, em que se considera o ponto inicial como sendo aquele que apresenta o valor mais baixo de chave primária.

Tendo por base os valores retornados pela *query* anterior, efetua-se uma nova *query* à mesma tabela, desta vez com o objetivo de retornar os valores de chave primária para todos os *track_ points* compreendidos entre o inicial e o final, resultando num *array* contendo todos os valores de chave primária dos *track_ points* necessários para a construção da geometria do segmento.

De seguida, o conteúdo deste *array* é inserido na tabela *track_points_largura* a qual tem caráter auxiliar, conjuntamente com o respetivo *track_id*, ou seja, a chave estrangeira correspondente ao respetivo percurso. No primeiro e no último elemento do *array*, a *query* utilizada retorna o valor de chave primária atribuído ao *track_point* na presente tabela. O retorno destes valores é necessário, visto que a próxima operação implica a inserção do valor *timestamp* na respetiva coluna, apenas para os *track_points* previamente inseridos.

A geometria do segmento é construida através de uma *query* à tabela *track_points*, com recurso à função *ST_MakeLine* sobre as geometrias dos *track_points* correspondentes ao segmento. Esta função permite construir uma nova geometria, neste caso do tipo linha, a partir de um conjunto de geometrias de pontos.

O processo de criação de um segmento de percurso relativo à largura fica concluído, com a inserção da geometria previamente criada, em conjunto com o valor das chaves primárias do percurso correspondente, *track_points* inicial e final, assim como o *timestamp* da inserção.

Para o caso da criação de um segmento de percurso relativo ao tipo de piso, encontra-se implementada a mesma rotina descrita pelo diagrama de atividades da figura 3.9, apenas com uma modificação subtil, visto que existem múltiplos tipos de piso. O tipo de piso específico do segmento é determinado a partir do campo correspondente no objeto JSON de entrada, sendo também inserido em conjunto com os restantes dados, na última etapa do processo, conforme descrito para os segmentos relativos à largura, mas numa tabela diferente, neste caso denominada *segmentos_tipo_piso*. Existe também uma tabela auxiliar específica para guardar informação dos *track_points* correspondentes aos segmentos inseridos, denominada, neste caso, *track_points_tipo_piso*.

3.4.4 Pesquisa de Percursos

Esta funcionalidade encontra-se parcialmente implementada, por forma a demonstrar resultados de pesquisa com carácter compensatório, e não compensatório.

A pesquisa pode, de momento, ser efetuada através de duas modalidades, pesquisa básica, e pesquisa avançada. A primeira, consiste num pedido WFS *GetFeature* realizado diretamente do cliente ao *Geoserver*, o qual retorna todos os percursos existentes no sistema, sem qualquer filtragem. A segunda permite obter percursos de acordo com uma combinação de parâmetros de pesquisa definidos pelo utilizador. Este, pode efetuar uma pesquisa de carácter não compensatório, em que os resultados obtidos correspondem exclusivamente aos parâmetros escolhidos, ou uma pesquisa de carácter compensatório, em que são incluídos resultados que, não correspondendo exclusivamente aos parâmetros inseridos, aparecem ordenadamente, consoante uma ordem de preferência implícita. Nesta modalidade, a combinação de parâmetros introduzida pelo utilizador corresponde a uma *query* estática, presente no servidor. Assim sendo, o *ID* da referida *query* é determinado no cliente, sendo de seguida efetuado um pedido ao *scipt pesquisa.php*, do qual faz parte o referido *ID*.

A figura 3.10 representa um diagrama de atividades correspondente à rotina do referido *script*, conforme este se encontra, de momento.

Antes da execução de qualquer *query*, é criada uma *view* contendo todos os dados essenciais a qualquer tipo de pesquisa, nomeadamente, chaves primárias e nomes de todos os percursos, assim como uma contagem de ocorrências, por cada percurso, correspondentes a cada tipo de segmento, obstrução ou ponto de interesse associados a cada percurso. A *view* é criada com recurso a uma sequência de vários *left join lateral*, os quais podem ser vistos como um ciclo *foreach* em SQL, o qual itera sobre cada linha retornada pela *query* primária, aplicando uma *subquery* que utiliza a referida linha como parâmetro [39]. Neste caso, a referida *subquery* aplica a função *count* nas tabelas correspondentes, tendo em conta o tipo de

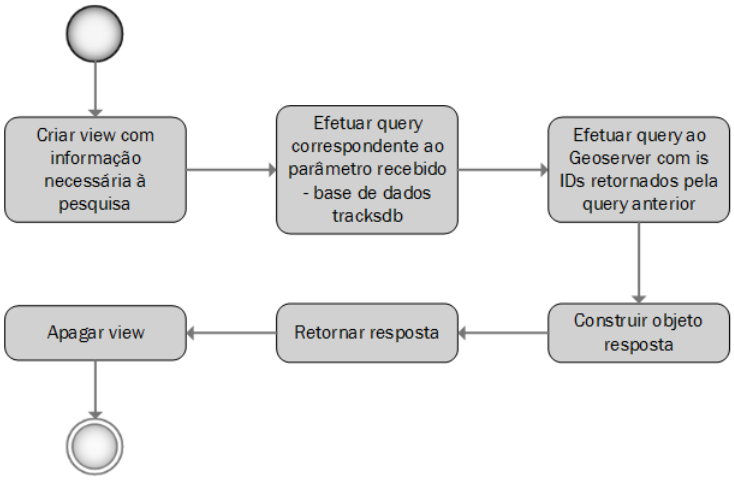

Figura 3.10: Diagrama de atividades correspondente à sequência de operações realizadas em *pesquisa.php*

segmento, obstrução e ponto de interesse, por cada percurso. A figura 3.11 representa uma vista parcial da referida *view*.

track_id integer	nome character	largura bigint	terra bigint	gravilha bigint
368	Aveirott	1	1	1
369	Track #1	3	0	2
370	Track #1	0	3	0
371	Track #2	2	0	3
372	Track #3	9	0	4
373	Track #1	0	2	0
374	Track #2	2	0	0
375	Track #3	1	0	0
376	Track #1	0	4	0
377	Enduro Ág	0	5	1

Figura 3.11: Vista parcial da *view* criada para a realização de pesquisas de percursos.

A partir desta *view*, podem ser retornados os *IDs* dos percursos correspondentes aos parâmetros de pesquisa definidos pelo utilizador. No caso de se efetuar uma *query* pesquisa com carácter não compensatório em que se pretenda excluir percursos onde existam um tipo particular de segmento, obstrução ou ponto de interesse, são retornados os *IDs* para os quais o resultado da contagem das referidas ocorrências é zero. Na versão compensatória desta pesquisa, são incluídos outros resultados, desta vez ordenados de forma ascendente, indicando assim uma ordem de preferência relativa ao número de ocorrências por segmento.

Uma vez obtidos os *IDs* dos percursos correspondentes aos parâmetros de pesquisa, é efetuado um pedido WFS *GetFeature* ao *Geoserver*, via *cURL*, onde são incluídos os referidos *IDs*. Este, retorna um objeto GeoJSON, correspondente a uma *FeatureCollection*, a qual contem um conjunto de *Features* que por sua vez contém a geometria e respetivas propriedades dos percursos a retornar ao cliente.

A operação final deste *script* consiste na criação de um objeto JSON, o qual contém dois elementos distintos. No primeiro, é incluído o resultado da *query* de pesquisa, efetuada à *view*, contendo os *IDs*, e respetivos nomes dos percursos. Esta informação destina-se ao preenchimento da tabela de resultados, no cliente, a partir da qual, o utilizador poderá consultar

a lista ordenada dos percursos, retornada de acordo com os parâmetros de pesquisa, podendo também selecionar os mesmos a partir desta tabela. O segundo elemento corresponde à *FeatureCollection* retornada pelo *Geoserver*. Esta informação destina-se ao mapa. A geometria de cada percurso é renderizada como *overlay*, sendo que as respetivas propriedades podem ser consultadas em *popup*, ativado através de clique sobre a geometria de cada percurso. A figura 3.12 representa um exemplo da resposta de uma operação de pesquisa de percursos, onde é visível um percurso selecionado.

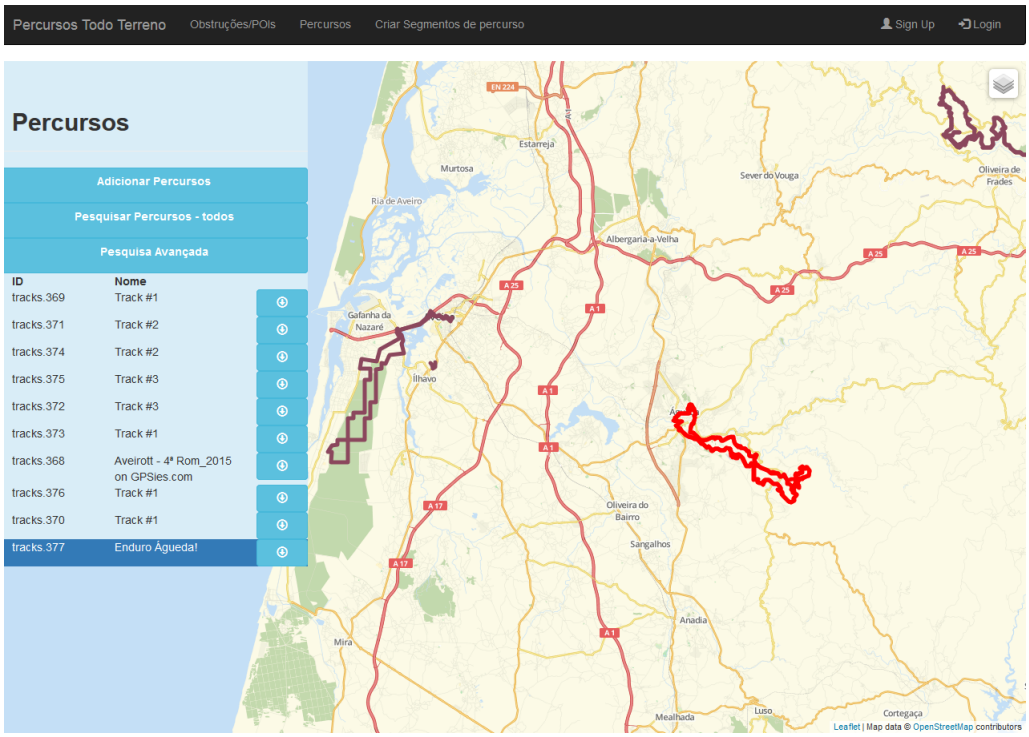

Figura 3.12: Vista do resultado de uma operação de pesquisa

A pesquisa avançada de percursos inclui ainda uma sub-funcionalidade parcialmente implementada, tendo esta por objetivo, permitir que o utilizador descarregue para o sistema de ficheiros do seu dispositivo, os percursos resultantes da pesquisa efetuada. Para tal, cada campo existente na tabela de resultados inclui um botão para o efeito, o qual inicia um pedido GET via *ajax*, que inclui o *ID* do percurso que o utilizador pretende descarregar. A resposta do servidor correspondente a este pedido corresponde a uma URL, que indica a localização do respetivo ficheiro GPX, no servidor. A referida URL é incluída num elemento HTML gerado dinamicamente, aquando da receção da mesma no cliente. De momento, este processo termina, com a disponibilização do conteúdo do respetivo ficheiro no *browser*, numa nova página.

A figura 3.13 mostra um diagrama de atividades referente ao processo de descarregamento de percursos existentes no sistema.

O pedido GET é efetuado a um *script* existente no servidor, designado por *download_ track.php*, que tem por finalidade a conversão dos percursos existentes na base de dados, para ficheiros GPX, os quais são armazenados numa pasta contida no diretório do servidor. O nome atribuído a cada um destes ficheiros, ao serem criados, é composto pelo prefixo *tracks_* seguido do *ID* do percurso.

A primeira operação efetuada pelo *script* consiste em verificar se existe algum ficheiro correspondente ao percurso que se pretende descarregar. Para tal, recorre-se à função *file_ exists*,

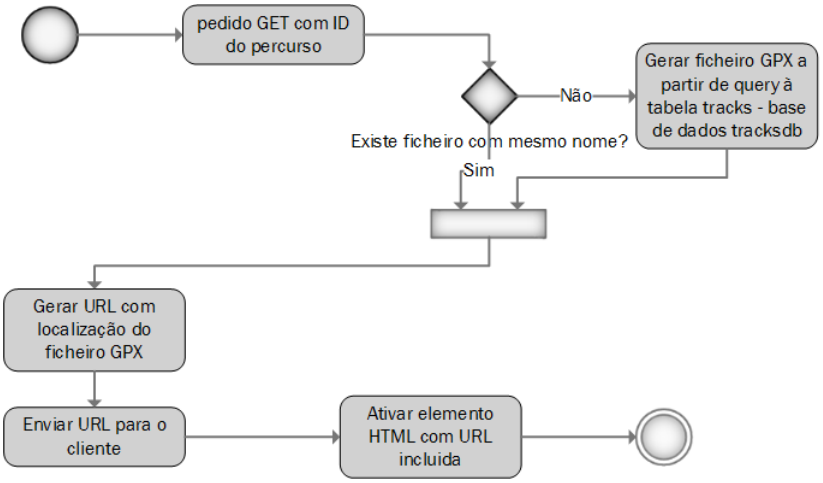

Figura 3.13: Diagrama de atividades correspondente ao processo de descarregamento de percursos.

do PHP, assim como ao valor de *ID* recebido como parâmetro do pedido GET. Caso exista, procede-se de imediato à criação da URL que aponta para a localização do respetivo ficheiro no diretório do servidor, retornando-a, de seguida, para o cliente.

Caso contrário, procede-se à conversão do percurso correspondente ao *ID* recebido, com recurso à ferramenta *ogr2ogr*, resultando num ficheiro GPX, que é armazenado na referida pasta, contendo o respetivo percurso, gerando-se, de seguida a respetiva URL, e consequente resposta ao cliente.

Capítulo 4

Desenvolvimento da Solução

No capítulo anterior foi feita uma descrição dos tipos de dados que a aplicação desenvolvida permite gerir e disponibilizar, assim como das funcionalidades responsáveis pela inserção, manipulação, e disponibilização dos referidos dados, conforme se encontram implementadas neste momento.

Este capitulo tem por objetivo abordar questões relativas ao processo de conceção da solução proposta, nomeadamente, opções tomadas relativamente à distribuição dos serviços e funcionalidades no sistema, dificuldades encontradas, e respetiva solução encontrada, assim como uma breve reflexão sobre cada uma destas questões.

4.1 Visão Geral da Arquitetura

O sistema desenvolvido envolve envolve a aplicação de soluções já existentes para a disponibilização de serviços, assim como algumas funcionalidades desenvolvidas de raiz. O *Geoserver*, o *OSM Tile Server* e o *backend* POI-DP representam as ditas soluções já existentes, neste caso para disponibilização ao cliente de percursos, cartografia base e inserção de POIs, respetivamente. Optou-se pela distribuição destes por três máquinas distintas.

A primeira destas máquinas alberga o *Geoserver*, a base de dados dos percursos, e disponibiliza *scripts* relacionados com as funcionalidades de inserção de percursos, criação de segmentos, e pesquisa de percursos. A figura 4.1 mostra os componentes do sistema presentes nesta máquina.

O *Geoserver* encarrega-se de gerar as respostas aos pedidos de pesquisa de percursos, vindos do cliente. Estas respostas encontram-se estruturadas no formato GeoJSON, o qual é diretamente representável visualmente, sobre a cartografia base, com recurso à biblioteca *Leaflet*.

Os *scripts* visíveis na figura 4.1 são servidos pelo servidor web *Apache* localizado nesta máquina. Foram desenvolvidos para efetuar operações de inserção e pesquisa de percursos, assim como criação de segmentos de percurso.

O serviço de cartografia base, assim como o cliente da aplicação são disponibilizados a partir de uma máquina distinta. A figura 4.2 representa os componentes relevantes da referida máquina.

O *Apache Web Server* implementado nesta máquina encarrega-se de servir a aplicação cliente, a qual se distribui pelos elementos conectados ao servidor *Apache*, conforme representado na figura 4.2. Os ficheiros *map.html*, *script.js* e *style.css* correspondem à vertente cliente da aplicação desenvolvida. Os restantes elementos representados correspondem às bibliotecas, e respetivas extensões utilizadas.

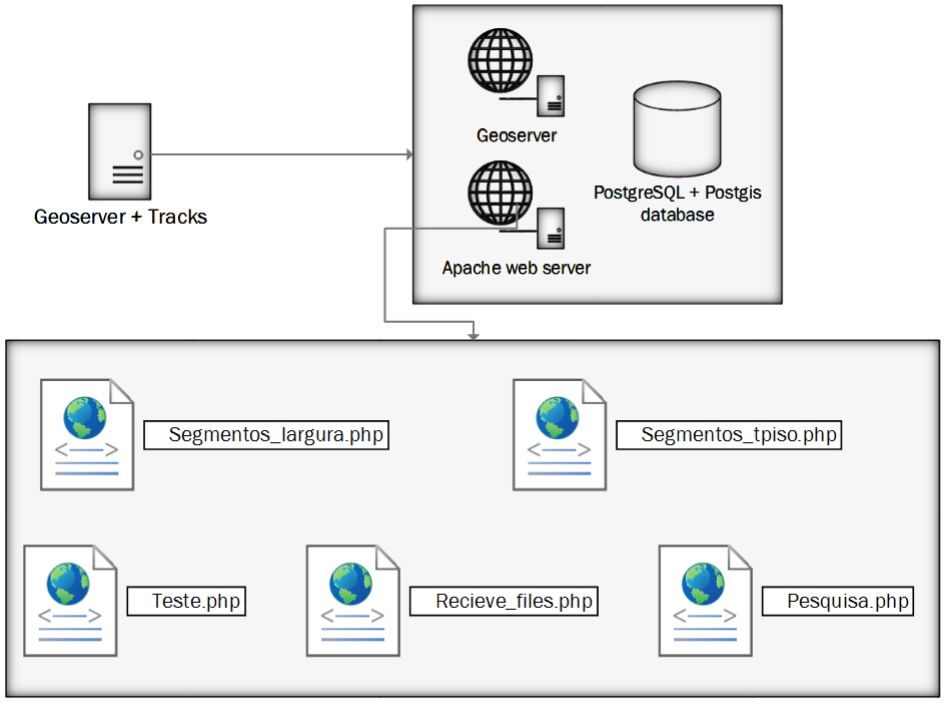

Figura 4.1: Vista dos componentes do sistema presentes na máquina Geoserver-tracks

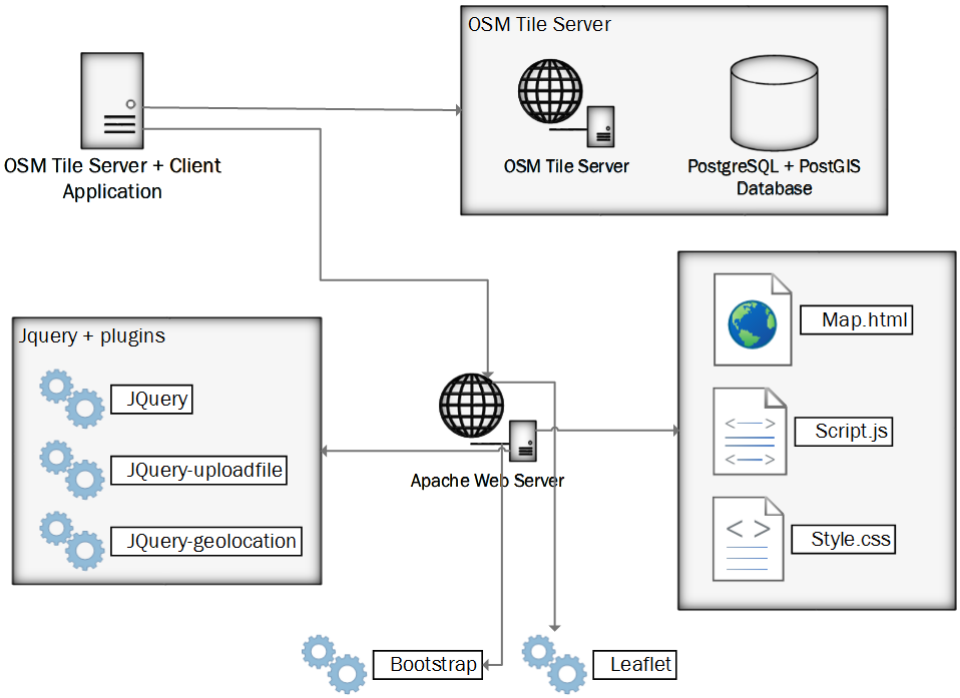

Figura 4.2: Vista dos componentes do sistema presentes na máquina OSM-tiles

O *OSM Tile Server* funciona de forma paralela ao serviço da aplicação cliente. Este será abordado com maior detalhe noutra secção deste capítulo.

Por fim, o *backend* POI-DP, em conjunto com dois *scripts* relacionados com as funcionalidades de inserção de POIs e criação de segmentos de percurso, são disponibilizados, também por uma máquina distinta. A figura 4.3 representa os componentes relevantes da mesma.

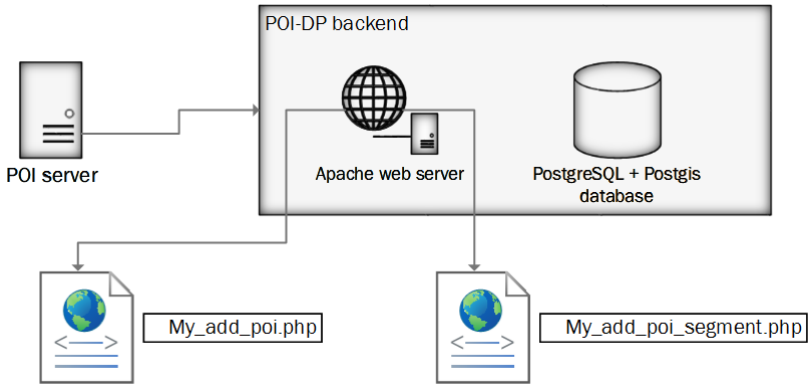

Figura 4.3: Vista dos componentes do sistema presentes na máquina POI Server

Os dois *scripts* representados na figura foram adicionados ao *backend* POI-DP. Cada um deles diz respeito a uma funcionalidade distinta, conforme foi descrito no capítulo anterior, em que ambas utilizam este *backend*.

A opção por uma arquitetura distribuída, com múltiplos servidores, em que cada um deles apresenta finalidades distintas tem em vista a distribuição de carga de processamento, por forma de melhorar o desempenho da aplicação, especialmente quando esta se encontre a lidar com múltiplos pedidos provenientes de múltiplos utilizadores. Para além disto, pretende-se, com esta arquitetura, construir um sistema com uma relativa flexibilidade em relação a falhas. Em alternativa a uma arquitetura em que o sistema se encontre implementado numa única máquina, esta arquitetura permite manter a aplicação parcialmente operacional, caso uma das máquinas fique *offline*, exceto no caso da máquina que serve a aplicação cliente ficar indisponível. Uma situação possível seria o caso em que a máquina que alberga o *backend* POI-DP se encontre *offline*. Nesta situação não seria possível inserir inserir obstruções e pontos de interesse, ou segmentos de percurso. No entanto, a inserção de percursos estaria funcional. Relativamente à pesquisa de percursos, esta ficaria limitada, visto que não haveria acesso à informação sobre obstruções e pontos de interesse. Seria, no entanto, possível pesquisar percursos, utilizando os segmentos de percurso como critério.

4.2 Armazenamento de Dados

Conforme ficara explícito no capítulo anterior, a informação é armazenada em duas bases de dados distintas. Ambas implementadas com recurso ao SGBD *PostgreSQL*, e espacialmente ativadas pela extensão *PostGIS*. Uma delas, designada por *tracksdb*, armazena toda a informação relacionada com percursos e segmentos de percursos, ou seja, informação cuja representação geográfica corresponde a linhas. A segunda, a qual integra o *backend* POI-DP, armazena toda a informação pontual existente no sistema, ou seja, obstruções, pontos de interesse, assim como os POIs relacionados com a criação dos segmentos de percurso. Esta base de dados tem a designação *poidatabase*.

4.2.1 *tracksdb*

Esta base de dados armazena a informação relativa aos percursos e segmentos de percurso. A sua estrutura baseia-se no resultado da conversão dos ficheiros GPX para PostgreSQL, através da ferramenta *ogr2ogr*, à qual foram adicionadas novas tabelas, destinadas ao armazenamento de informação sobre os segmentos de percurso. A figura 4.4 representa um modelo lógico desta base de dados.

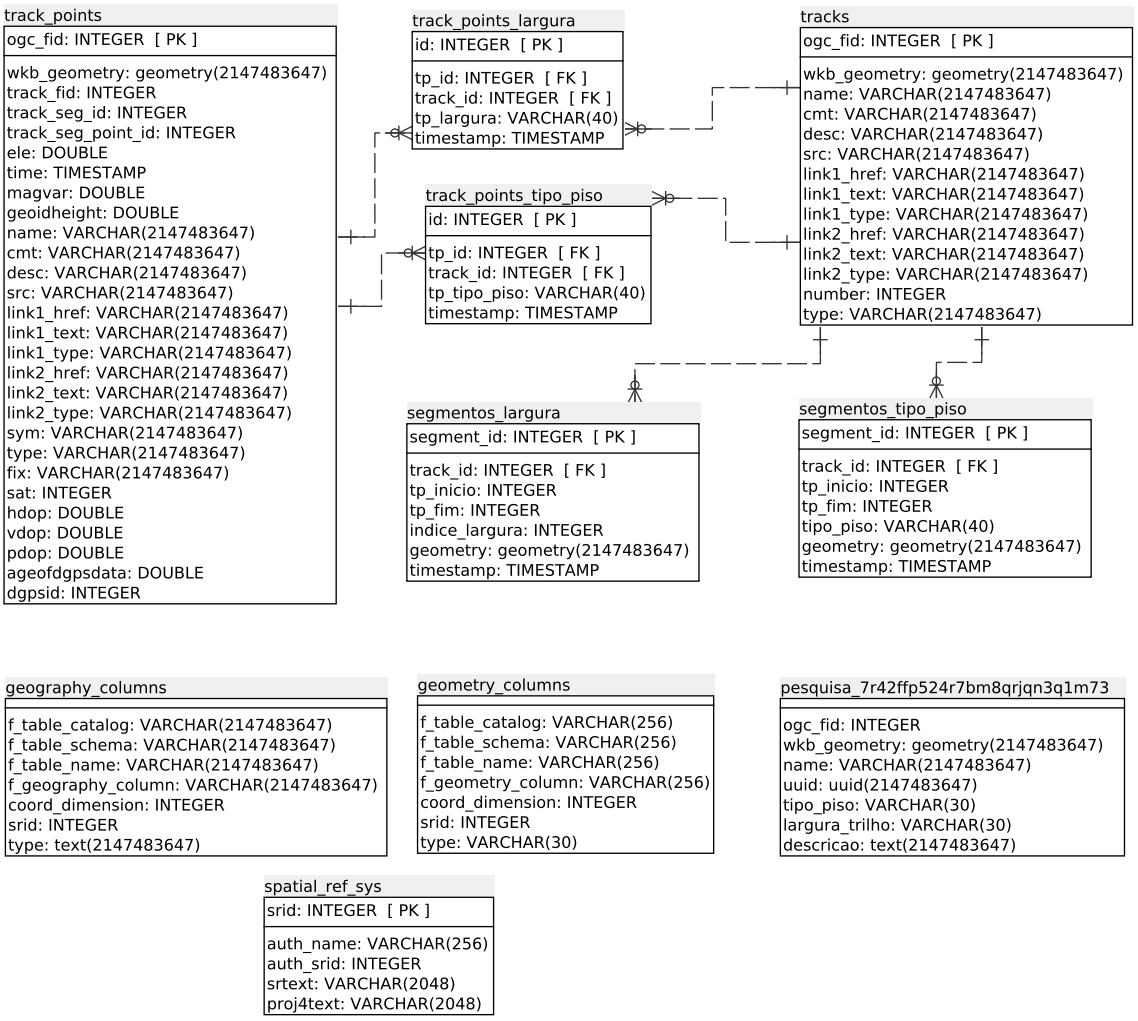

Figura 4.4: Modelo lógico da base de dados *tracksdb*

geography_ columns e *geometry_ columns* correspondem a *views*, estas, em conjunto com a tabela *spatial_ ref_ sys* são criadas por defeito aquando da ativação espacial da base de dados, pela extensão *PostGIS*. As duas primeiras estão relacionadas com os tipos de dados definidos pela extensão, geografias e geometrias, respetivamente. A tabela *spatial_ ref_ sys* guarda os códigos dos sistemas de referência espaciais existentes.

As tabelas *tracks* e *track_ points* guardam a informação referente aos percursos. Estas tabelas foram reutilizadas, sem qualquer modificação, do *output* da conversão dos ficheiros GPX para PostgreSQL, através da ferramenta *ogr2ogr*.

As tabelas *track_ points_ largura* e *track_ points_ tipo_ piso* foram criadas, com objetivo de auxiliar a criação de segmentos de percurso, para cada tipo de segmento. Estas estabelecem a relação entre conjuntos distintos de *track_ points* e o respetivo *track*, associando-se ainda a informação que identifica o tipo de segmento.

As *tabelas segmentos_ largura* e *segmentos_ tipo_ piso* guardam a geometria dos segmentos, os *IDs* dos *track_ points* inicial e final do segmento, informação temporal referente ao momento em que cada segmento foi criado, e estabelecem, ainda, relação com o percurso correspondente, através do atributo *track_ id*.

Foi incluída também no modelo a *view* criada dinamicamente para as pesquisas de percursos. Esta contém toda a informação necessária para efetuar a pesquisa, combinando percursos, segmentos e POIs. A informação dos POIs é obtida através de uma ligação remota à base de dados do *backend* POI-DP. Esta é assegurada por uma extensão do SGBD *PostgreSQL*, denominada *postgres_fdw*, a qual corresponde a um *Foreign Data Wrapper* (FDW). Os FDW representam um método extensível e estandardizado que permite ao SGBD interrogar outras fontes de dados [40].

4.2.2 *poidatabase*

Destinada ao armazenamento de obstruções, pontos de interesse e POIs associados à criação de segmentos de percurso, esta base de dados apresenta a estrutura definida para o componente *fw_core*, definido pela arquitetura POI-DP, à qual foi adicionada uma nova tabela, para armazenar informação extra. A figura 4.5 representa um modelo lógico desta base de dados.

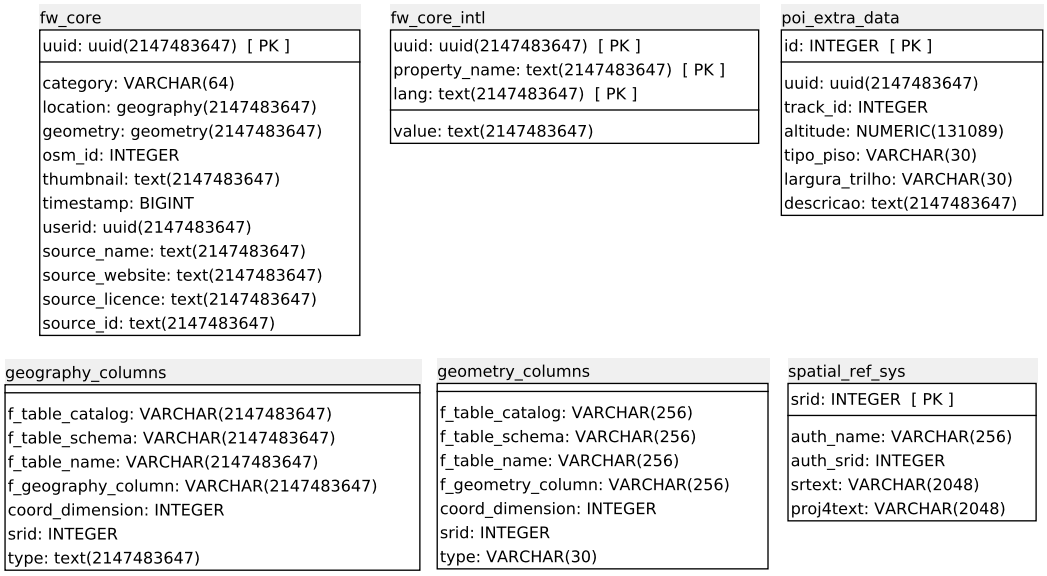

Figura 4.5: Modelo lógico da base de dados *poidatabase*

As tabelas *fw_core* e *fw_core_intl* compõe o referido componente *fw_core*. A primeira, armazena informações importantes, entre as quais o identificador universal do POI, designadamente *uuid*, a categoria, o e a respetiva representação espacial, tanto geométrica, assim como geográfica. A tabela *poi_extra_data* foi adicionada para estabelecer relação entre cada POI e o respetivo percurso correspondente, por intermédio dos *IDs* dos mesmos. Armazena também informações que definem o tipo de POI. Os atributos *altura* e *descrição* foram também incluídos, no entanto não tem ainda utilização definida no presente estado de desenvolvimento da aplicação.

Existe uma questão relativa à criação desta última tabela, visto que a informação nela armazenada poderia, em vez disso, ser armazenada na tabela *fw_core_intl*, recorrendo exclusivamente à API do *backend* POI-DP, eliminando a necessidade de manipular o a estrutura do POI no servidor. Esta questão será novamente abordada no capítulo final deste documento.

4.3 Instalação do OSM Tile Server

A disponibilização de cartografia base poderia ser assegurada através de fornecedores comerciais de mapas, tais como *Google*, *Bing*, entre outros, ou através do serviço de cartografia base OSM. No entanto, optou-se pela implementação de uma servidor de cartografia base dedicado, o qual disponibiliza, de momento, a mesma cartografia OSM. Esta opção baseou-se na questão da utilização da aplicação com recurso a dispositivos móveis, particularmente, numa aplicação móvel a desenvolver futuramente, capaz de recolher dados em modo *offline*.

A instalação e configuração do *OSM Tile Server* foi efetuada de acordo com as instruções disponibilizadas no *Website switch2osm*, incluindo a configuração da folha de estilos *OSM Bright*, cujas instruções encontram-se também incluidas. [26].

O *dataset* destinado a popular a base de dados corresponde à área geográfica de Portugal continental, e arquipélago da Madeira, o qual é disponibilizado para descarregamento a partir do *Website download.geofabrik.de*. A figura 4.6, retirada da mesma fonte, representa a referida área geográfica. [41]

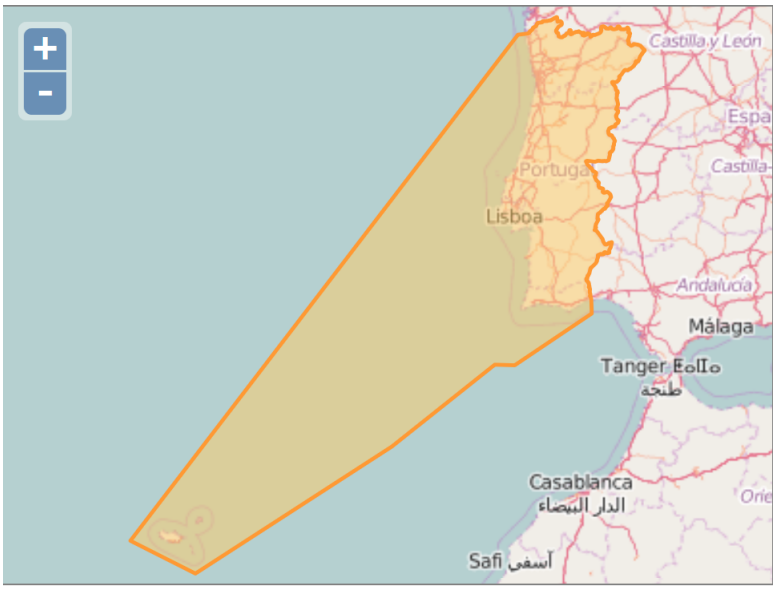

Figura 4.6: Área geográfica do *dataset* presente na base de dados do *OSM Tile Server*

4.4 Carregamento de Percursos

Inicialmente, foi testada a ferramenta *ogr2ogr*, existente na biblioteca GDAL, para efetuar o carregamento de dados para a base de dados. Trata-se de uma ferramenta capaz de converter dados de, e para múltiplos formatos de dados espaciais, para além de proporcionar outras funcionalidades. Para a conversão de formato GPX para PostgreSQL, conforme pretendido, a ferramenta é invocada, a partir da linha de comandos, indicando-se como parâmetro o formato de saída, neste caso PostgreSQL, seguido dos respetivos parâmetros de ligação à base de dados, e por fim, o caminho para o ficheiro GPX que se pretende carregar.

Como resultado, assumindo-se que a ferramenta executou na normalidade, são criadas cinco novas tabelas na base de dados, com uma estrutura correspondente ao modelo lógico representado na figura 4.7.

Estas tabelas representam camadas, em que cada uma delas corresponde a um dos tipos de objeto geográfico definido para o esquema GPX. São criadas por defeito, independentemente

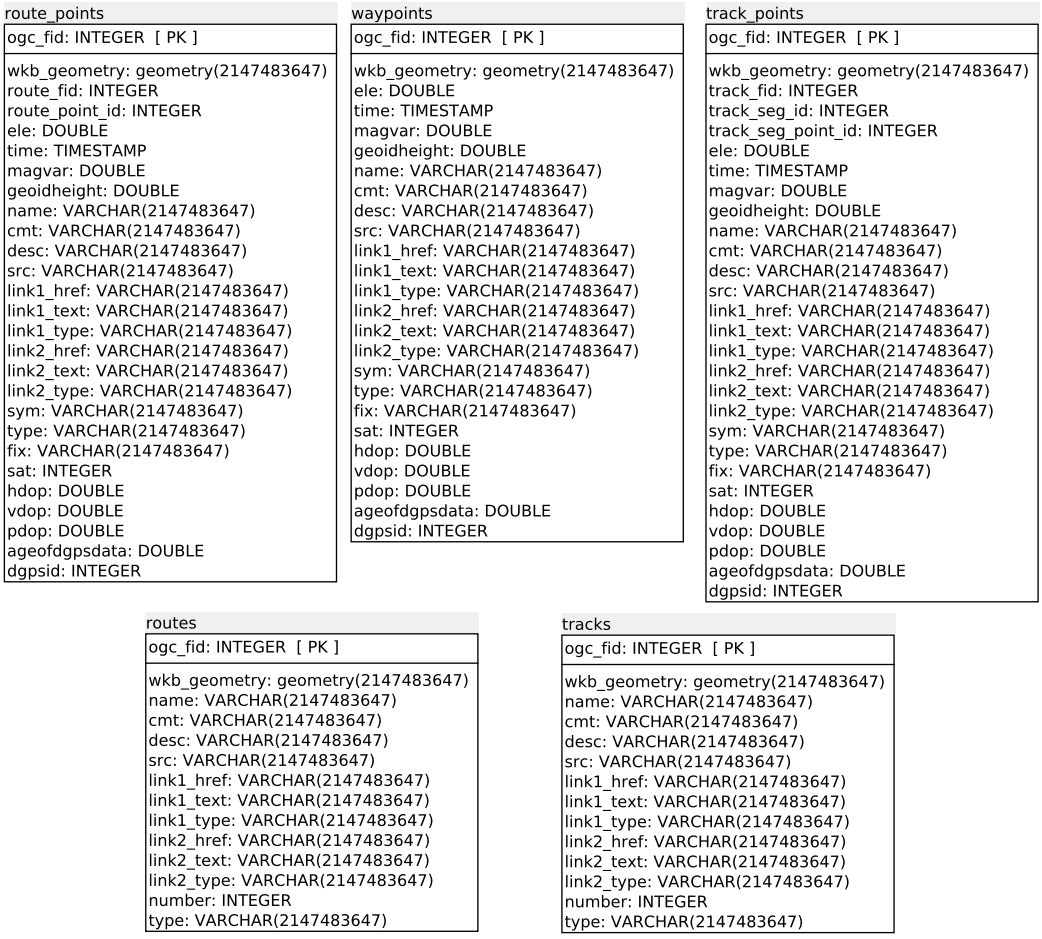

Figura 4.7: Modelo lógico da base de dados *trackstempdb*

de o ficheiro conter ou não todos os tipos de objeto referidos anteriormente. Os ficheiros utilizados para teste, por exemplo, continham apenas as camadas *tracks* e *track_ points*, em que, a primeira armazena objetos geográficos do tipo linha, correspondentes ao trajeto dos percursos, e a segunda, objetos do tipo ponto, que correspondem a leituras pontuais de coordenadas, aquando do levantamento do percurso.

Considerando que a ferramenta ogr2ogr permite acrescentar dados às camadas já existentes, através da inclusão do parâmetro *append*, entendeu-se que a estrutura de tabelas criada pela ferramenta ogr2ogr seria adequada para armazenar persistentemente os dados de percursos, pelo que foi testada a adição de dados de outros ficheiros GPX às mesmas tabelas, o que acabou por originar complicações.

Foram testados dois ficheiros com características diferentes: o primeiro (ficheiro A) contém apenas um percurso, e o segundo (ficheiro B) contém três percursos.

Após vários testes, verificou-se que, quer para A ou B, os dados de um ficheiro são introduzidos corretamente, e é estabelecida relação entre ambas as tabelas. O recurso à ferramenta ogr2ogr funciona bem, assumindo que a base de dados contenha apenas dados provenientes de um ficheiro.

Ao adicionar um segundo ficheiro, ocorre um erro de SQL, devido à tentativa de introdução de um valor de chave primária já existente, pois o tipo de dados é serial. Ao que se pode verificar, a ferramenta ogr2ogr parece assumir sempre, mesmo em caso de *append*, que a base

de dados destino está vazia, e inicia sempre uma nova série de valores para a chave primária, ao carregar um ficheiro, originando assim o erro, para o caso em que a base de dados já contém dados.

Este problema inviabiliza o recurso exclusivo à ferramenta ogr2ogr para carregar dados de ficheiros GPX para a base de dados PostgreSQL. No entanto, para não descartar o recurso a esta ferramenta, assim como a estrutura de tabelas por ela criada, foi proposta uma solução com um pouco mais de complexidade, envolvendo duas bases de dados, sendo uma temporária e outra persistente, e manipulação dos dados através de um *script* em PHP.

Nesta solução, cada ficheiro é carregado com recurso à ferramenta ogr2ogr, conforme descrito nos parágrafos anteriores, sendo que neste caso, para a base de dados temporária, cujo modelo lógico se encontra representado na figura 4.7. A partir daqui, os dados são importados da base de dados temporária, para a base de dados persistente. Permite-se assim controlar a inserção de novos percursos, para que não haja conflitos com a chave primária, e que seja preservada a relação entre os dados das duas tabelas. Os seguintes parágrafos contém uma descrição mais detalhada de como os dados são importados do ficheiro GPX para a base de dados persistente.

Foi desenvolvido um *script* em PHP, o qual, após estabelecer ligação a ambas as bases de dados, executa a ferramenta ogr2ogr, criando assim as tabelas na base de dados temporária, e preenchendo-as com os dados do ficheiro recebido.

De forma a preservar a integridade da base de dados persistente, os dados das tabelas *tracks* e *track_points* são inseridos nesta de forma sequencial, sendo que, por cada entrada da tabela *tracks* inserida, são, de seguida, selecionadas e introduzidas apenas as entradas da tabela *track_points* correspondentes à referida entrada de *tracks*. Logo após a inserção de uma *track*, é necessário saber o valor da chave primária da mesma, pois esse valor terá de ser inserido manualmente na tabela *track_points*, como chave estrangeira.

Existem, pelo menos, três formas de obter este valor, tendo sido testada a função *currval* conjuntamente com a função *pg_get_serial_sequence*. Esta alternativa é viável, visto que as chaves primárias das tabelas são geradas através de sequências. A primeira função retorna o último valor obtido por outra função, neste caso *nextval*, para a sequência que se encontra em sessão. A segunda função retorna o nome da sequência em sessão, visto que se trata de um parâmetro obrigatório para a função *currval*. Note-se que a função *nextval* é executada automaticamente, quando se efetua uma inserção de dados na tabela. Mesmo quando múltiplas sessões executam a função *nextval* concorrentemente, cada uma delas recebe um valor de sequência distinto, de forma segura. A segunda alternativa seria o recurso à função *lastval*, a qual limita-se a retornar o valor de *nextval* da sequência mais recentemente modificada. A terceira opção implica incluir uma cláusula *returning* na query de inserção, a qual retorna o valor inserido numa coluna especificada, que neste caso seria a coluna correspondente à chave primária.

Pelo que se apurou até ao momento, a terceira alternativa, descrita no parágrafo anterior, seria preferível, pois não origina qualquer tipo de conflito. Existem algumas questões relativas às restantes soluções, concretamente para o caso de existir uma regra ou *trigger* na base de dados que implica, aquando da inserção de dados numa tabela, sejam feitas inserções extra de dados noutras tabelas. Nestes casos, a alternativa da função *lastval* não deverá retornar o valor da sequência correto, o mesmo podendo acontecer com a função *currval*, caso as inserções extras sejam feitas na própria tabela, embora seja pouco usual. No entanto, o recurso à função *currval* para obter o valor da chave primária do último elemento introduzido, aparentemente, encontra-se a funcionar corretamente, razão pela qual optou-se por manter esta alternativa.

Após a inserção de um percurso na tabela *tracks*, na base de dados persistente, e da respetiva obtenção da chave primária do mesmo, procede-se à inserção dos pontos correspondentes

ao mesmo, na tabela *track_points*. Os referidos pontos são inseridos da mesma forma que os percursos, no entanto, é necessário preencher manualmente a coluna da chave estrangeira, para cada conjunto de pontos inserido. Por esta razão, é necessário, antes que seja inserido qualquer ponto, conhecer o valor de chave primária do último elemento inserido nesta tabela, para que sejam modificados apenas os valores correspondentes aos pontos inseridos no momento, mantendo assim intactos os valores de chave estrangeira de pontos inserido anteriormente. Para obtenção deste valor, foi reutilizada a alternativa da função *currval*, aplicada anteriormente para obter o valor da chave primária do percurso, na tabela *tracks*.

Neste último passo, duas situações distintas podem ocorrer. Se não for obtido qualquer valor de chave primária, significa que a tabela se encontra vazia. Desta forma, a coluna da chave estrangeira é atualizada incondicionalmente, visto que todos os pontos existentes, nesta situação, correspondem a um único percurso. Caso tenha sido obtido um valor de chave primária, significa que já existem pontos inseridos anteriormente. Assim, apenas os elementos cujo valor de chave estrangeira seja superior ao valor obtido antes da inserção serão atualizados.

Aqui, o recurso à função *currval* revelou-se ser uma opção errada, pois a mesma encontra-se dependente do retorno da função *nextval* na corrente sessão, a qual por sua vez ocorre automaticamente, após uma inserção. No caso dos pontos, seria necessário a sua chamada antes da inserção, pelos motivos mencionados anteriormente. Os testes que incluíram esta função para obter a chave primária do último registo de *track_points* revelaram um bug, em que, na primeira iteração do ciclo de inserção, a atualização da coluna da chave estrangeira comportava-se sempre como se a tabela se encontrasse vazia, ou seja, independentemente de já terem sido inseridos percursos e respetivos pontos, a coluna da chave estrangeira era sempre atualizada incondicionalmente. Tal devia-se ao facto de, na primeira iteração do ciclo de inserção, nunca ser possível obter qualquer valor para a chave primária do último registo de *track_points*, visto que, conforme descrito anteriormente, ainda não teria ocorrido inserção de dados. Para quaisquer outras iterações do ciclo, este problema já não ocorre para as restantes iterações, pois já terão previamente ocorrido inserções de dados.

Assim sendo, a solução encontrada foi uma *query* de seleção aos valores da coluna da chave primária, com ordenação descendente, e limitados a apenas um valor, o que permite retornar sempre o último valor de chave primária da tabela, caso exista. Os testes com esta solução originaram os resultados pretendidos, não se verificando qualquer problema.

Este procedimento é repetido até que todos os percursos e respetivos pontos contidos na base de dados temporária sejam importados para a base de dados persistente. Por fim, as tabelas geradas pela ferramenta *ogr2ogr*, na base de dados temporária, são apagadas, visto já não serem necessárias, e, aquando do carregamento de um novo ficheiro GPX, serão novamente geradas pela ferramenta.

4.5 Criação de Segmentos de Percurso

A primeira abordagem de implementação desta funcionalidade baseou-se na criação de segmentos através da submissão de um único POI. Partiu-se do pressuposto de que, cada percurso tem princípio e fim, em que estes são definidos pelo menor e maior valor de chave primária dos *track_points* correspondentes, e, o POI submetido corresponderia, exclusivamente, ao início ou fim de um segmento.

No caso da submissão de um POI correspondente ao início de um segmento, o sistema assumiria que o segmento pretendido engloba todos os *track_points* compreendidos entre o *track_point* mais próximo do POI marcado, e o fim do percurso. Caso o POI submetido correspondesse ao fim de um segmento, o sistema assumiria, então, que o respetivo segmento

englobava todos os *track_points* compreendidos entre o início do percurso, e o *track_point* correspondente ao POI inserido.

Esta abordagem de implementação revelou-se ineficaz, por duas razões. Em primeiro lugar, para o caso da criação de um segmento compreendido entre dois *track_points* no interior do percurso. Neste caso específico, o utilizador teria obrigatoriamente de inserir dois POIs, para definir a extensão pretendida do segmento. Isto representa um novo problema de implementação, em que, por exemplo, o segmento teria de ser criado em duas etapas. Após a marcação de uma das extremidades do segmento, e respetiva criação do mesmo, conforme estabelecido por esta abordagem, este teria de ser retornado ao cliente, de forma equiparada a um percurso. Nesta primeira etapa, consideraria-se um segmento parcial. Na segunda etapa, o utilizador poderia marcar a extremidade restante, truncando, desta forma o segmento parcial, e originando o verdadeiro segmento pretendido.

Em segundo lugar, existe outra questão relativa aos pontos iniciais e finais dos percursos, que torna inadequada esta abordagem de implementação. Considera-se que, do ponto de vista do utilizador, tanto o início como o fim do percurso, são arbitrários. Embora o levantamento do percurso tenha começado numa determinada localização, e terminado noutra, o mesmo poderá ser percorrido em ambos os sentidos. Assim sendo, o utilizador deverá considerar as referidas extremidades do percurso indiferentemente do valor de chave primária ou do *timestamp* do respetivo *track_point*. O sistema apenas teria uma forma de determinar a direção por onde se estende o segmento ao longo do percurso, com base nas características do POI inserido, ou seja, para um POI correspondente ao início de um segmento, este seria criado em direção ao *track_point* de maior valor de chave primária no percurso, o qual é visto como fim do percurso. Em grande parte dos casos, não iria corresponder ao pretendido pelo utilizador, visto que, do seu ponto de vista, os pontos iniciais e finais considerados são arbitrários, logo, poderá considerar o início do percurso em qualquer das extremidades, existindo, desta forma, uma grande possibilidade de o segmento ser criado com a direção contrária à pretendida.

Embora se considere que esta última questão poderia ser contornada, disponibilizando sobre o mapa os pontos inicial e final do percurso, indicando, desta forma, a ordem correta de marcação dos pontos inicial e final do segmento, optou-se por uma mudança de estratégia, em que a criação de um segmento obriga à marcação do ponto inicial, seguido do ponto final, e só posteriormente esta informação é encaminhada para o servidor, através de um único pedido. Desta forma, o segmento de percurso, para qualquer das situações descritas nos parágrafos anteriores, é sempre definido pelo conjunto de *track_points* compreendidos entre os POIs recebidos do cliente.

4.6 Pesquisa Avançada

O funcionamento da pesquisa de percursos tem por objetivo a criação, de forma dinâmica, de uma *query* de pesquisa, por cada operação de pesquisa, consoante os parâmetros definidos pelo utilizador, retornando, de seguida, um conjunto de resultados, ordenados segundo os respetivos parâmetros, sendo estes disponibilizados ordenadamente numa tabela de resultados, assim como renderizados sobre o mapa. Ao longo do desenvolvimento desta funcionalidade, constatou-se que esta implementação seria de complexidade superior, em relação às restantes funcionalidades. Por esta razão, optou-se por uma solução intermédia, capaz de obter resultados que demonstrem que a implementação proposta seja viável.

Assim sendo, o botão *pesquisa avançada*, presente no cliente, disponibiliza um formulário, o qual disponibiliza oito opções de pesquisa avançada, das quais apenas uma é submetida para o servidor. A figura 4.8 demonstra o referido formulário, tal como se encontra implementado de momento.

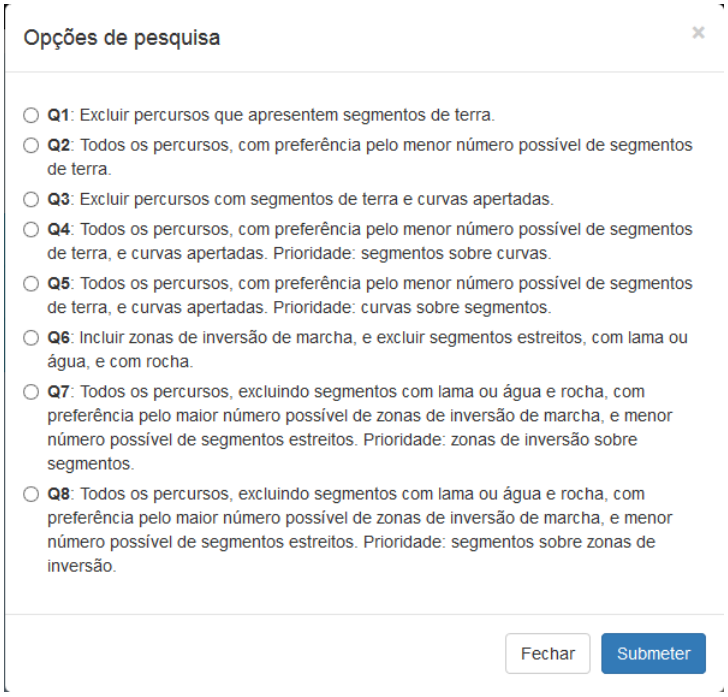

Figura 4.8: Vista do formulário de pesquisa avançada, com as opções de pesquisa presentemente implementadas.

Conforme referido anteriormente, cada uma destas opções de pesquisa, representa uma *query* estática, implementada no *script pesquisa.php*. Estas, representam combinações de parâmetros possíveis, tendo em conta os tipos de informação com que o sistema pode lidar, atualmente.

As opções *Q1* e *Q2* representam uma situação em que o utilizador pretende pesquisar percursos que não apresentem segmentos do tipo terra. A primeira opção apresenta uma pesquisa de carácter não compensatório, em que, no conjunto de resultados retornados, não constam percursos com qualquer segmento do tipo terra. A segunda opção, representa a vertente compensatória da mesma pesquisa. No conjunto de resultados retornados, aparecem, em primeiro lugar, todos os percursos que não contém qualquer segmento do tipo terra, seguidos de conjuntos de percursos que, embora apresentem segmentos do tipo terra, encontram-se ordenados de forma ascendente, consoante o número de segmentos deste tipo, presentes em cada percurso.

As opções *Q3*, *Q4* e *Q5* representam uma hipótese de pesquisa, em que se encontram envolvidos dois parâmetros. Neste caso são pretendidos percursos em que, para além de não conterem segmentos do tipo terra, não devem conter, também curvas apertadas. Assim como *Q1*, a opção *Q3* efetua uma pesquisa não compensatória, excluindo todos os percursos que contenham os referidos segmentos, assim como curvas apertadas. As restantes opções efetuam a vertente compensatória desta pesquisa, de duas formas distintas. Em *Q4* o interesse em percursos com o menor número possível de segmentos sobrepõe-se aos percursos com o menor número possível de curvas apertadas. Desta forma, os referidos percursos são ordenados, em primeiro lugar de acordo com o número de segmentos de terra, aplicando-se posteriormente a ordenação ascendente consoante o número de curvas apertadas. Caso existam dois o mais percursos com o mesmo número de segmentos de terra, estes, serão ordenados, relativamente a eles próprios, do menor para o maior número de curvas existentes.

As restantes três opções representam uma situação semelhante, a qual envolve a combi-

nação de um maior número de parâmetros de pesquisa. As opções *Q7* e *Q8* apresentam uma combinação de pesquisa compensatória, com pesquisa não compensatória. Nestes casos, os parâmetros água ou lama, e rocha representam critérios de exclusão obrigatória dos resultados, correspondendo à vertente não compensatória desta pesquisa. No entanto, é efetuada uma ordenação, ascendente para os segmentos de largura reduzida, e descendente para zonas de inversão de marcha, correspondendo à vertente compensatória desta pesquisa. Desta forma, são retornados todos os percursos, com o menor número possível de segmentos apertados, e maior número possível de zonas de inversão de marcha, em primeiro lugar, não constando dos resultados nenhum percurso que apresente segmentos de lama ou água, ou rocha.

Os resultados das operações de pesquisa, para além de serem renderizados sobre o mapa, são também disponibilizados numa tabela de resultados, na qual são incluídos os atributos *ID* e *nome* dos percursos. Os resultados de uma operação de pesquisa, presentes nesta tabela, devem encontrar-se dispostos pela mesma ordem obtida pela *query* à *view* de pesquisa, para o caso de uma operação de pesquisa de caráter compensatório.

Inicialmente, aquando da implementação da pesquisa básica, esta tabela era preenchida com informação das propriedades das *Features* correspondentes aos percursos, existentes na *FeatureCollection* retornada pelo *Geoserver*. No entanto, nesta fase não existia ainda a questão da pesquisa avançada, com parâmetros personalizados, pelo que a ordenação dos resultados não foi tida em conta. Neste caso, o pedido WFS *GetFEature* efetuado ao *Geoserver* limita-se a retornar todos os elementos presentes na *view pt:tracks*, existente no servidor, a qual disponibiliza a informação contida na tabela *tracks* da base de dados *tracksdb*, ou seja, todos os percursos inseridos no sistema. As *Features* encontram-se ordenadas na *FeatureCollection* segundo o respetivo *ID*, de forma ascendente.

O serviço WFS *GetFeature* implementado pelo *Geoserver* suporta a utilização de filtros *Extended Common Query Language* (ECQL) como parâmetro, que permitem obter uma seleção personalizada de *Features*, de acordo com as opções do filtro incluido como parâmetro do pedido *GetFEature* [42] [42]. No entanto, na referência ECQL [42] não constam opções de filtragem relacionadas com a ordenação de resultados, pelo que, pode-se concluir que o *Geoserver* por si só, não tem capacidade para retornar resultados, ordenados de acordo com os parâmetros de uma pesquisa de carácter compensatório.

Por esta razão, a resposta do *script pesquisa.php* consiste num objeto JSON, constituído por dois elementos distintos. O primeiro corresponde ao resultado da *query* à *view* de pesquisa, os quais encontram-se ordenados de acordo com os parâmetros escolhidos pelo utilizador. O segundo corresponde à *FeatureCollection* retornada pelo *Geoserver*, em função dos *IDs* inseridos como parâmetros do pedido *GetFeature*.

Uma vez recebido no cliente, os IDs e nomes dos percursos contidos no primeiro elemento do objeto JSON são utilizados para o preenchimento da tabela de resultados, preservando-se, desta forma, a ordenação estabelecida pela operação de pesquisa, e as *Features* contidas no segundo elemento deste objeto, as quais contém as geometrias dos percursos conjuntamente com as respetivas propriedades, destinam-se à renderização destes percursos sobre a cartografia base, com recurso à biblioteca *Leaflet*.

Capítulo 5

Resultados

Ao longo deste capítulo são expostos os resultados dos testes às funcionalidades implementadas. Estes foram efetuados com recurso a apenas um cliente a interagir com o sistema, tendo por objetivo verificar se as funcionalidades produzem os resultados pretendidos.

Não foi, até ao momento testado sistematicamente o cenário em que múltiplos clientes acedem ao sistema, e efetuam operações simultaneamente, nem o cenário em que um dos servidores se encontra *offline*.

As primeiras duas secções abordam os testes realizados à inserção de percursos e criação de segmentos. Uma vez que a criação de segmentos engloba a inserção de POIs, entende-se que os testes realizados a esta funcionalidade são também válidos para a inserção de obstruções e pontos de interesse. Pretende-se verificar a integridade dos dados, após a inserção dos mesmos. A última secção aborda os testes realizados à pesquisa de percursos. Pretende-se verificar se o resultado retornado corresponde aos parâmetros inseridos.

5.1 Inserção de Percursos

A figura 5.1 representa um excerto de um dos ficheiros GPX utilizados para testes. Na estrutura de dados representada, constata-se que o percurso em questão contém o respetivo nome, assim como um conjunto de pontos, os quais definem o traçado do percurso. Cada ponto tem atribuídas as respetivas coordenadas, contendo ainda dois atributos associados a este, elevação e tempo. Note-se que as datas representadas no atributo *time* dos metadados não coincide com as do atributo correspondente, nos pontos. Trata-se provavelmente de uma questão de definições do dispositivo utilizado na captura do percurso, não se considerando relevante para a análise dos resultados.

A interface cliente disponibiliza um formulário de inserção de ficheiros, o qual pode ser acedido através de um botão existente no menu *Percursos*, que por sua vez é acedido a partir da barra de navegação. A figura 5.2 mostra um exemplo de carregamento de ficheiros para o sistema.

O formulário utilizado permite carregar múltiplos ficheiros, podendo estes ser simplesmente arrastados para o interior da caixa, ou acedidos através do navegador de ficheiros local, pelo botão *Carregar*, iniciando-se, de forma automática, o carregamento. Note-se que o formulário informa o utilizador do facto de a aplicação apenas suportar o formato de ficheiros GPX para esta operação.

A barra de progresso indica o estado de carregamento para cada ficheiro a submeter. Enquanto a submissão do ficheiro se encontra em progresso, esta pode ser abortada, mediante ação do utilizador. Quando esta termina, A barra de progresso indica 100%, sendo precedida de um botão nomeado *Done*, situação esta que significa uma submissão bem sucedida.

```
<metadata>
  <name>Aveirott - 4ª Rom_2015</name>
  <link href="http://www.gpsies.com/">
    <text>Aveirott - 4ª Rom_2015 on GPSies.com</text>
  </link>
  <time>2015-06-26T01:35:12Z</time>
</metadata>
<trk>
  <name>Aveirott - 4ª Rom_2015 on GPSies.com</name>
  <trkseg>
    <trkpt lat="40.63865611" lon="-8.64382938">
      <ele>0.0000000</ele>
      <time>2010-01-01T00:00:00Z</time>
    </trkpt>
    <trkpt lat="40.63774914" lon="-8.64420588">
      <ele>0.0000000</ele>
      <time>2010-01-01T00:00:38Z</time>
    </trkpt>
```

Figura 5.1: Excerto de um ficheiro GPX

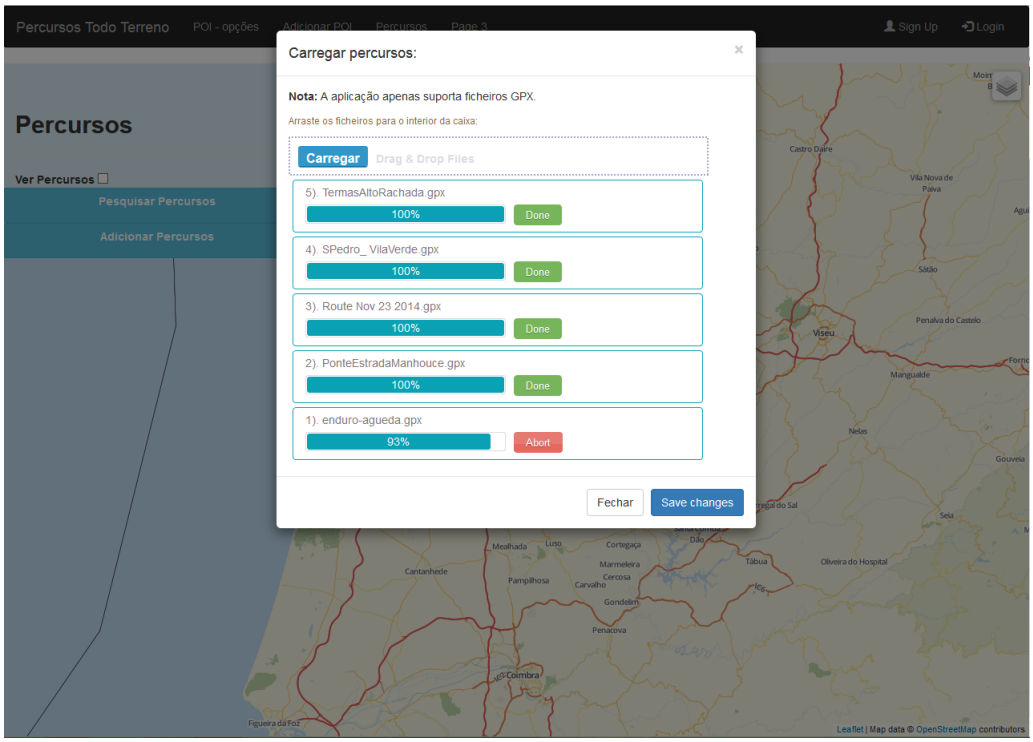

Figura 5.2: Vista do formulário de carregamento de percursos

Após inserções bem sucedidas, a informação contida nos ficheiros fica armazenada na base de dados, conforme representado na figura 5.3, para a tabela *tracks*, e na figura 5.4, para a tabela *track_points*.

Os valores da coluna geometria visíveis nas figuras 5.3 e 5.4 foram obtidos com recurso a uma função espacial da biblioteca *PostGIS*, nomeadamente *ST_AsEWKT*, a qual retorna a informação armazenada nas colunas de geometria, em formato texto. Observando a primeira linha da tabela *tracks*, representada na figura 5.3, constata-se que o nome do percurso coincide com o nome do percurso no ficheiro GPX, representado na figura 5.1, assim como as coordenadas do ponto correspondente à primeira linha da tabela *track_points* correspondem com o

	ogc_fid integer	st_asewkt text	name character varying
1	368	SRID=4326;MULTILINESTRING((-8.64382938 40.63865611,-8.64420588 40.63774	Aveirott - 4ª Rom 2015 on GPSies.com
2	369	SRID=4326;MULTILINESTRING((-8.2155123 40.7726423,-8.2155043 40.7728109,	Track #1
3	370	SRID=4326;MULTILINESTRING((-8.0638093 40.7627189,-8.0639648 40.7627392,	Track #1
4	371	SRID=4326;MULTILINESTRING((-8.0997777 40.7428676,-8.0997455 40.7427701,	Track #2
5	372	SRID=4326;MULTILINESTRING((-8.2152387 40.7645961,-8.215338 40.7649211,-	Track #3
6	373	SRID=4326;MULTILINESTRING((-8.0638093 40.7627189,-8.0639648 40.7627392,	Track #1
7	374	SRID=4326;MULTILINESTRING((-8.0997777 40.7428676,-8.0997455 40.7427701,	Track #2
8	375	SRID=4326;MULTILINESTRING((-8.2152387 40.7645961,-8.215338 40.7649211,-	Track #3
9	376	SRID=4326;MULTILINESTRING((-8.6615074 40.6103566,-8.6607939 40.6105521,	Track #1
10	377	SRID=4326;MULTILINESTRING((-8.453074 40.572952,-8.453022 40.572922,-8.4	Enduro Águeda!

Figura 5.3: Vista da informação contida na base de dados *tracks*

	ogc_fid integer	st_asewkt text	track_fid integer	time timestamp with time zone
1	410023	SRID=4326;POINT(-8.64382938 40.63865611)	368	2010-01-01 00:00:00+00
2	410024	SRID=4326;POINT(-8.64420588 40.63774914)	368	2010-01-01 00:00:38+00
3	410025	SRID=4326;POINT(-8.64428223 40.63770638)	368	2010-01-01 00:00:40+00
4	410026	SRID=4326;POINT(-8.64442093 40.63767415)	368	2010-01-01 00:00:45+00
5	410027	SRID=4326;POINT(-8.64461218 40.63773482)	368	2010-01-01 00:00:51+00
6	410028	SRID=4326;POINT(-8.64496207 40.63801905)	368	2010-01-01 00:01:07+00
7	410029	SRID=4326;POINT(-8.64538786 40.63836384)	368	2010-01-01 00:01:26+00
8	410030	SRID=4326;POINT(-8.64544526 40.63844212)	368	2010-01-01 00:01:29+00
9	410031	SRID=4326;POINT(-8.64546387 40.63851142)	368	2010-01-01 00:01:32+00

Figura 5.4: Vista da informação contida na base de dados *track_points*

primeiro par de coordenadas visível na geometria do percurso que aparece em primeiro lugar da tabela *tracks*, e também com as coordenadas do primeiro ponto do percurso contido no ficheiro, o que permite concluir que a informação geográfica contida no ficheiro foi preservada na íntegra. Constata-se ainda que, para as mesmas linhas descritas, o valor de chave primária de *tracks* coincide com o valor de *track_fid*, o que demonstra que a relação entre os percursos na tabela *tracks* com os respetivos pontos, na tabela *track_points* está a ser estabelecida corretamente.

5.2 Criação de Segmentos de Percurso

O teste efetuado a esta funcionalidade consistiu na criação de quatro segmentos, no mesmo percurso, em que os dois primeiros correspondem a segmentos do tipo *gravilha* e *terra* respetivamente, o terceiro corresponde a um segmento de largura estreita, e o último corresponde a um segmento do tipo *lama/água*. Os segmentos referidos foram criados por esta ordem.

A figura 5.5 mostra uma vista do estado da aplicação cliente, no final da criação de um segmento. É visível uma linha a roxo sobre a cartografia base, a qual corresponde a um percurso carregado a partir da base de dados. Os percursos carregados na interface cliente, sobre a cartografia base, servem de confirmação às constatações efetuadas na secção anterior. É visível, também, um elemento *alert* no centro da interface, o qual é representado em maior detalhe na figura 5.6. Este elemento ocorre no final da inserção do segmento, e contém a resposta do *backend* POI-DP, relativamente à inserção dos POIs referentes ao segmento.

Aquando de uma inserção bem sucedida no *backend* POI-DP, este retorna um objeto JSON, como resposta ao correspondente pedido, contendo o identificador do POI inserido, assim como o *timestamp* da inserção. Na figura 5.6 são visíveis dois destes objetos, os quais correspondem aos POIs que definem o início e o fim do segmento. Neste momento é possível constatar que as rotinas implementadas na vertente cliente da aplicação funcionam corretamente, visto que ambos os POIs são inseridos com sucesso no *backend* POI-DP.

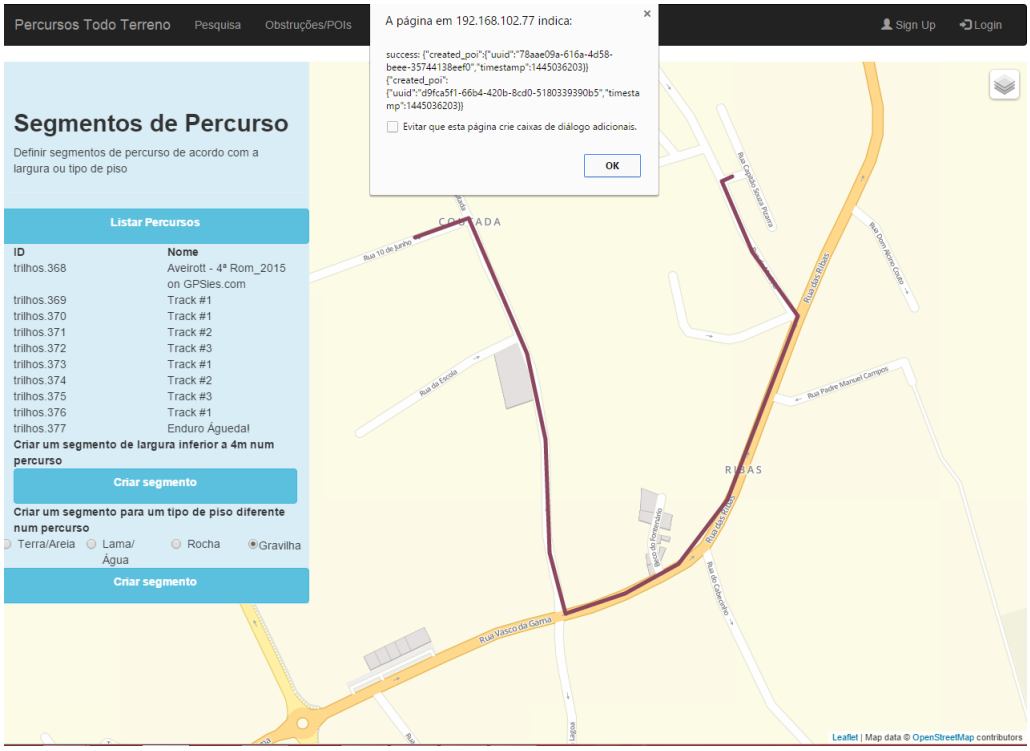

Figura 5.5: Vista do fim do processo de criação de segmentos no cliente

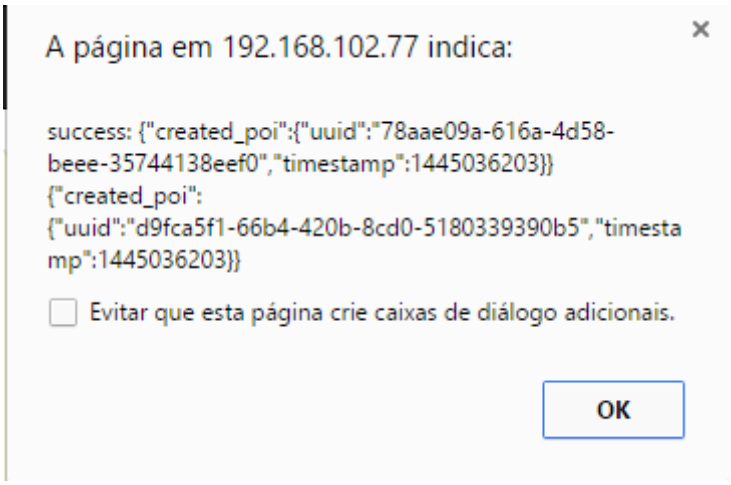

Figura 5.6: Aproximação do elemento *alert* contendo a resposta do *backend* POI-DP

As figuras 5.7 e 5.8 representam os POIs armazenados na base de dados do *backend* POI-DP, após uma inserção bem sucedida dos mesmos.

Constata-se que os POIs inseridos no *backend* POI-DP foram inseridos com a categoria correta, a relação com o percurso foi também estabelecida corretamente, e, por fim, a ordem pela qual os POIs se encontram na base de dados, corresponde à ordem por qual os segmentos foram criados.

A figura 5.9 representa o conteúdo das tabelas *segmentos_tipo_piso*, em cima, e *segmentos_largura* em baixo. Constata-se que os segmentos foram introduzidos nas tabelas corretas, o identificador do percurso correspondente foi preservado, a geometria do segmento foi criada com sucesso, e, por fim, o *timestamp* foi também inserido com sucesso.

	uuid uuid	category character v	st_asewkt text	timestamp bigint
1	87ff0b0f-fc52-403f-9959-fc2dcedd61d2	segmentos	SRID=4326;POINT(-8.6848211288452 40.620141896098)	1446183863
2	e1180a07-1c7e-43f0-ace1-7d1bfe7e0328	segmentos	SRID=4326;POINT(-8.6878252029419 40.616135103883)	1446183863
3	6551be58-76f6-4b29-aab0-5743e6b95a59	segmentos	SRID=4326;POINT(-8.7073516845703 40.598834730678)	1446183906
4	5954aead-097e-4d05-aa33-ea82ba311fef	segmentos	SRID=4326;POINT(-8.7093257904053 40.584365444046)	1446183906
5	558fd92f-b704-4d5f-952b-342fd7d3d158	segmentos	SRID=4326;POINT(-8.7205696105957 40.613366042017)	1446183943
6	59295573-f512-424b-8229-d20baacbee9f	segmentos	SRID=4326;POINT(-8.7222003936768 40.606133362688)	1446183943
7	424b8371-ddfb-48fc-a017-e7e4a909bbb8	segmentos	SRID=4326;POINT(-8.7468338012695 40.546808911919)	1446183991
8	2ca48a8a-eafc-4371-a4bf-059b7a1afb97	segmentos	SRID=4326;POINT(-8.7306976318359 40.546026260077)	1446183992

Figura 5.7: Vista da informação contida na tabela *fw_ core*

	id integer	uuid uuid	track_id integer	tipo_piso character varying(30)	largura_trilho character varying(30)
1	87	87ff0b0f-fc52-403f-9959-fc2dcedd61d2	368	gravilha-inicio	
2	88	e1180a07-1c7e-43f0-ace1-7d1bfe7e0328	368	gravilha-fim	
3	89	6551be58-76f6-4b29-aab0-5743e6b95a59	368	terra-inicio	
4	90	5954aead-097e-4d05-aa33-ea82ba311fef	368	terra-fim	
5	91	558fd92f-b704-4d5f-952b-342fd7d3d158	368		largura-inicio
6	92	59295573-f512-424b-8229-d20baacbee9f	368		largura-fim
7	93	424b8371-ddfb-48fc-a017-e7e4a909bbb8	368	lama-agua-inicio	
8	94	2ca48a8a-eafc-4371-a4bf-059b7a1afb97	368	lama-agua-fim	

Figura 5.8: Vista da informação contida na tabela *poi_ extra_ data*

segment_id integer	track_id integer	tp_inicio integer	tp_fim integer	tipo_piso character varying(40)	st_asewkt text	timestamp timestamp with time zone
5	368	410296	410308	segmento-gravilha	SRID=4326;MULTILINESTRING((-8.68725015 40.61636387,	2015-10-30 05:44:26.653033+00
6	368	410272	410275	segmento-terra	SRID=4326;MULTILINESTRING((-8.70802778 40.58480433,	2015-10-30 05:45:08.384517+00
7	368	410246	410251	segmento-lama-agua	SRID=4326;MULTILINESTRING((-8.74619583 40.54746549,	2015-10-30 05:46:34.194565+00

segment_id integer	track_id integer	tp_inicio integer	tp_fim integer	indice_largura integer	st_asewkt text	timestamp timestamp with time zone
9	368	410201	410209	0	SRID=4326;MULTILINESTRING((-8.72033947 40.6129882,	2015-10-30 05:45:45.424451+00

Figura 5.9: Vista da informação contida nas tabelas *segmentos_ tipo_ piso* e *segmentos_ largura*

5.3 Pesquisa de Percursos

Para efetuar os testes a esta funcionalidade, foram inseridas algumas obstruções, pontos de interesse e segmentos, em mais alguns percursos, de forma a que estes apresentem números variáveis de POIs e segmentos associados. A figura 5.10 representa algumas das colunas da *view* a partir da qual são efetuadas as pesquisas de percursos. Note-se que existem dez percursos inseridos no sistema, cujos *IDs* variam entre 368 e 377.

	track_id integer	nome character v	largura bigint	terra bigint	gravilha bigint	lama_agua bigint
1	368	Aveirott	1	2	1	1
2	369	Track #1	3	0	2	0
3	370	Track #1	0	4	0	0
4	371	Track #2	2	0	3	2
5	372	Track #3	9	0	4	0
6	373	Track #1	0	2	0	3
7	374	Track #2	2	0	0	0
8	375	Track #3	1	0	0	1
9	376	Track #1	0	4	0	0
10	377	Enduro Águ	0	5	1	4

Figura 5.10: Vista parcial da view *pesquisa*

Esta *view* cruza informação da tabela *tracks*, nomeadamente contagens dos segmentos,

categorizada por tipo de segmento, associados a cada percurso, com informação da tabela *fw_ core*, nomeadamente contagens das obstruções e pontos de interesse, também categorizadas por tipo, por cada percurso.

Os testes efetuados a esta funcionalidade pretendem demonstrar se os resultados retornados por cada operação de pesquisa se encontram em conformidade com os parâmetros da respetiva *query*. As seguintes operações de pesquisa são abordadas nesta secção:

- **Q3:** Excluir percursos com segmentos de terra, e curvas apertadas.
- **Q4:** Todos os percursos, com preferência pelo menor número possível de segmentos de terra, e curvas apertadas. Prioridade: segmentos sobre curvas.
- **Q5:** Todos os percursos, com preferência pelo menor número possível de segmentos de terra, e curvas apertadas. Prioridade: curvas sobre segmentos.

Q3 corresponde a uma operação de pesquisa de caráter não compensatório, pelo que deve retornar, como resultado, todos os percursos existentes, cuja contagem para segmentos de terra, assim como para curvas apertadas, seja igual a zero. A figura 5.11 mostra uma seleção da *view* de pesquisa, onde foram incluídas apenas as contagens de segmentos de terra, e curvas apertadas, para cada percurso. Tendo em conta os dados presentes nesta seleção, constata-se que os percursos 369, 371 e 374 correspondem aos parâmetros da operação de pesquisa Q3.

	track_id integer	nome character varying	terra bigint	curva bigint
1	368	Aveirott - 4ª Rom 2015	2	5
2	369	Track #1	0	0
3	370	Track #1	4	3
4	371	Track #2	0	0
5	372	Track #3	0	4
6	373	Track #1	2	2
7	374	Track #2	0	0
8	375	Track #3	0	1
9	376	Track #1	4	1
10	377	Enduro Águeda!	5	3

Figura 5.11: Seleção da *view* de pesquisa, com as contagens de segmentos de terra, e curvas apertadas.

A figura 5.12 mostra o resultado desta operação de pesquisa na tabela de resultados. Constata-se que os percursos retornados ao cliente correspondem aos parâmetros da operação de pesquisa Q3.

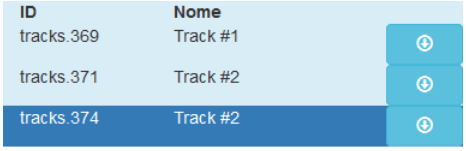

Figura 5.12: Vista da tabela de resultados,após a operação de pesquisa Q3.

Q4 corresponde a uma versão de caráter compensatório da operação de pesquisa Q3. Esta operação retorna todos os percursos disponíveis no sistema, no entanto, estes devem aparecer dispostos, na tabela de resultados, com uma ordenação ascendente, relativamente à contagem do número de segmentos de terra, e de curvas apertadas. Neste caso, a ordenação atribui prioridade aos segmentos de terra, sobre as curvas apertadas.

Os resultados previstos para a operação de pesquisa Q4 devem conter, em primeiro lugar, os mesmos percursos retornados pela operação Q3, visto serem estes os percursos cujas propriedades apresentam melhor correspondência com os parâmetros da operação de pesquisa. De seguida, devem constar os percursos 375 e 372, respetivamente. Respeitando a prioridade dos segmentos de terra sobre as curvas apertadas, ambos não apresentam segmentos de terra, no entanto, o percurso 375 apresenta uma curva apertada, enquanto que o percurso 372 apresenta quatro curvas apertadas. Os restantes percursos devem aparecer segundo esta lógica de ordenação, conforme é possível verificar na figura 5.13, a qual representa uma seleção dos percursos existentes na *view* de pesquisa, incluindo as contagens de segmentos de terra e curvas apertadas, em que os mesmos aparecem ordenados segundo os parâmetros de pesquisa da operação Q4.

	track_id integer	nome character varying	terra bigint	curva bigint
1	369	Track #1	0	0
2	371	Track #2	0	0
3	374	Track #2	0	0
4	375	Track #3	0	1
5	372	Track #3	0	4
6	373	Track #1	2	2
7	368	Aveirott - 4ª Rc	2	5
8	376	Track #1	4	1
9	370	Track #1	4	3
10	377	Enduro Águeda!	5	3

Figura 5.13: Seleção da *view* de pesquisa, com as contagens de segmentos de terra, e curvas apertadas. Percursos ordenados segundo os parâmetros da operação de pesquisa Q4

A figura 5.14 mostra o resultado da operação de pesquisa Q4 na tabela de resultados. Constata-se que os resultados encontram-se dispostos pela mesma ordem demonstrada na figura 5.13, concluindo-se os resultados enquadram-se com os critérios de ordenação desta operação de pesquisa.

ID	Nome
tracks.369	Track #1
tracks.371	Track #2
tracks.374	Track #2
tracks.375	Track #3
tracks.372	Track #3
tracks.373	Track #1
tracks.368	Aveirott - 4ª Rom_2015 on GPSies.com
tracks.376	Track #1
tracks.370	Track #1
tracks.377	Enduro Águeda!

Figura 5.14: Vista da tabela de resultados,após a operação de pesquisa Q4.

Assim como Q4, a operação de pesquisa Q5 corresponde também a uma versão de caráter compensatório da operação de pesquisa Q3. No entanto, a prioridade de ordenação em Q5 é inversa, face a Q4, ou seja, os resultados são ordenados, em primeiro lugar face à contagem de curvas apertadas, e de seguida, face à contagem de segmentos de terra.

Posto isto, os resultados previstos para esta operação de pesquisa devem conter em primeiro lugar, à semelhança de Q4, os mesmos resultados da operação de pesquisa Q3, aos quais

sucedem os percursos 375 e 376, respetivamente. Ambos os percursos apresentam uma curva apertada, tratando-se do parâmetro prioritário de ordenação nesta operação de pesquisa. O percurso 375 é colocado na tabela antes do percurso 376, visto que o primeiro apresenta zero segmentos de terra, enquanto que o segundo apresenta quatro. Os restantes percursos devem aparecer segundo esta lógica de ordenação, conforme é possível verificar na figura 5.15, a qual representa uma seleção dos percursos existentes na *view* de pesquisa, incluindo as contagens de segmentos de terra e curvas apertadas, em que os mesmos aparecem ordenados segundo os parâmetros de pesquisa da operação Q5.

	track_id integer	nome character varying	terra bigint	curva bigint
1	369	Track #1	0	0
2	371	Track #2	0	0
3	374	Track #2	0	0
4	375	Track #3	0	1
5	376	Track #1	4	1
6	373	Track #1	2	2
7	370	Track #1	4	3
8	377	Enduro Águeda!	5	3
9	372	Track #3	0	4
10	368	Aveirott - 4ª Rc	2	5

Figura 5.15: Seleção da *view* de pesquisa, com as contagens de segmentos de terra, e curvas apertadas. Percursos ordenados segundo os parâmetros da operação de pesquisa Q5

A figura 5.16 mostra o resultado da operação de pesquisa Q5 na tabela de resultados. Constata-se que os resultados encontram-se dispostos pela mesma ordem demonstrada na figura 5.15, concluindo-se os resultados enquadram-se com os critérios de ordenação desta operação de pesquisa.

ID	Nome
tracks.369	Track #1
tracks.371	Track #2
tracks.374	Track #2
tracks.375	Track #3
tracks.376	Track #1
tracks.373	Track #1
tracks.370	Track #1
tracks.377	Enduro Águeda!
tracks.372	Track #3
tracks.368	Aveirott - 4ª Rom_2015 on GPSies.com

Figura 5.16: Vista da tabela de resultados,após a operação de pesquisa Q5.

O comportamento observado para as operações de pesquisa abordadas nesta secção é representativo das restantes operações implementadas no sistema. Uma vez que a lógica de implementação mantém-se para todas, apenas mudam os parâmetros, os resultados observados para as restantes operações de pesquisa, em termos de ordenação dos percursos na tabela de resultados, são idênticos, relativamente aos casos aqui abordados.

Capítulo 6

Conclusões

6.1 Resumo do Trabalho Efetuado

O presente documento propõe uma aplicação do tipo *web mapping*, que permite a inserção, pesquisa e disponibilização de dados sobre percursos Todo-Terreno, recolhidos pelos praticantes das iniciativas TTT, com recurso a dispositivos GPS. Para além disto, a aplicação proposta permite ainda a inserção de informações de valor acrescentado, relacionadas com os referidos percursos, que possibilitam a realização de pesquisas personalizadas de percursos, as quais vão de encontro às necessidades das equipas de socorro, tais como, Bombeiros e Proteção Civil.

A aplicação proposta utiliza tecnologias e plataformas de código aberto, tais como a plataforma FIWARE, para a gestão dos POIs, assim como o *Geoserver*, para a disponibilização de dados sobre os percursos, através de *web services*. Estas soluções garantem interoperabilidade de dados com outros sistemas.

As funcionalidades implementadas permitem o carregamento de percursos, inserção de informação de valor acrescentado sobre os percursos, nomeadamente, obstruções, pontos de interesse, e segmentos de percursos, realização de pesquisas de percursos, com base nas informações de valor acrescentado existentes, e descarregamento dos percursos resultantes das operações pesquisa.

A inserção de percursos foi inicialmente pensada de forma a incluir a capacidade de levantamento de percursos com recurso ao sistema de posicionamento do dispositivo, no entanto, implementação desta funcionalidade limitou-se ao carregamento de percursos contidos em ficheiros representados em formato padrão, obtidos por outras fontes. As informações extra sobre os percursos ficam asseguradas com recurso aos POIs. Estes, foram também utilizados na criação de segmentos de percurso, constituindo uma funcionalidade que, não estando bem explícita à partida, a sua utilidade tornou-se evidente numa fase mais avançada do desenvolvimento da aplicação.

A pesquisa de percursos encontra-se Implementada de forma provisória, devendo ser melhorada no futuro. Neste momento, encontra-se implementado no servidor, um conjunto de *querys* estáticas, ao qual corresponde, no cliente, uma lista de opções de pesquisa, em que cada opção aponta para a respetiva *query*. Trata-se de uma solução de caráter intermédio, destinada à realização de testes para avaliar as capacidades de pesquisa compensatória, e não compensatória.

6.2 Resultados Relevantes

As funcionalidades implementadas na aplicação desenvolvida permitem a inserção de percursos, POIs e segmentos com sucesso, conforme demonstrado, em particular, pelos testes realizados à funcionalidade de pesquisa. Constata-se, desta forma, que as soluções encontradas para os problemas relacionados com a inserção de percursos, nomeadamente o recurso a uma base de dados temporária, de forma a tirar partido da ferramenta *ogr2ogr*, funciona de forma satisfatória. É importante referir, no entanto, que a solução encontrada para garantir a integridade dos dados inseridos em contexto multi-utilizador, corresponde uma solução temporária, que deve ser substituída futuramente por uma solução a nível do SGBD.

A integração do POI-DP na aplicação desenvolvida constitui uma solução funcional para armazenar informações de valor acrescentado sobre os percursos inseridos no sistema, representando também potencial para futuros desenvolvimentos da aplicação, nomeadamente, novos componentes de dados relativos aos elementos pontuais considerados. Verifica-se também que a utilização de POIs na criação de segmentos constitui uma solução relativamente satisfatória, visto que permite obter porções de um percurso com sucesso, através de uma relação de proximidade entre o POI inserido pelo utilizador, e o *track_point* mais próximo do mesmo. Neste momento, os POIs são obtidos a partir de clique sobre o mapa, na interface do cliente. trata-se, novamente, de uma solução temporária, destinada á realização de testes. Para que seja tirado partido das funcionalidades que implicam a inserção de POIs, considera-se que a localização dos mesmos seja obtida no terreno, com recurso ao sistema de localização de um dispositivo móvel, garantindo, desta forma, uma exatidão posicional aceitável para os POIs inseridos.

Os resultados de pesquisa correspondem ao esperado, tanto na abordagem compensatória, como na não compensatória. Para o caso da abordagem compensatória, constata-se que os percursos aparecem ordenados, na tabela de resultados, sendo colocados em primeiro lugar aqueles que melhor se enquadram com os parâmetros definidos, sendo estes seguidos por percursos que se enquadram com os parâmetros, numa ordem sucessivamente inferior. No caso de uma ordenação compensatória baseada em mais que um parâmetro, na atual implementação, os resultados são ordenados de forma sistemática, de acordo com um parâmetro de cada vez. Isto significa que, para o caso de uma operação de pesquisa que pretenda retornar percursos com base no menor número possível de derrocadas e curvas, por exemplo, um percurso que contenha uma derrocada e seis curvas aparecerá primeiro que outro, com duas derrocadas e apenas uma curva, caso o parâmetro derrocada ocorra primeiro, na ordenação. Este exemplo, representa uma situação, em que o utilizador pode ter maior interesse no menor número possível de ocorrências de ambos os tipos, situação em que, neste momento, a ordenação compensatória não corresponde da melhor forma ao interesse do utilizador, uma vez que a operação de pesquisa retornaria uma ordenação inversa à pretendida.

Considera-se, em modo geral, que a aplicação desenvolvida, encontrando-se esta em fase de protótipo, demonstra potencial para servir os interesses dos praticantes de TTT, assim como proporcionar informações úteis sobre as características e estado dos percursos Todo-Terreno às equipas de socorro. No entanto, algumas das funcionalidades atualmente implementadas, assim como alguns aspetos de implementação devem ser remodelados, para suprimir eventuais problemas de desempenho e integridade resultantes das questões levantadas na análise de resultados. Soluções para as referidas questões levantadas encontram-se propostas na próxima secção do presente capítulo.

Além do mais, esta aplicação pode ser expandida com novas funcionalidades, que poderão potenciar a sua utilidade para ambos os tipos de utilizadores definidos no presente documento. A próxima secção aborda também algumas propostas de funcionalidades para implementação

futura.

6.3 Trabalho Futuro

6.3.1 Funcionalidades Implementadas e Aspetos de Implementação

Fica proposta a implementação de uma aplicação móvel, baseada na aplicação cliente aqui descrita, mas que acrescente a capacidade de recolha de dados em modo *offline*, em que os dados sejam guardados localmente, no dispositivo móvel, sendo posteriormente sincronizados com o servidor, por ação do utilizador, quando assim o pretender. Tal aplicação poderá ser desenvolvida com recurso à *framework* Apache Cordova, que permite o desenvolvimento de aplicações móveis para múltiplas plataformas, com recurso a tecnologias web [43]. A disponibilização de cartografia base não representaria grandes problemas, uma vez que os *tiles* da cartografia base podem ser descarregados para o dispositivo móvel, e utilizados também em modo *offline*. Visto que se encontra implementado um servidor dedicado para a disponibilização da cartografia base da OSM, contornando, desta forma as restrições existentes ao descarregamento de *tiles* de cartografia base a partir dos servidores públicos da OSM.

Relativamente a uma questão mencionada previamente a respeito do armazenamento de POIs no *backend* POI-DP no capítulo 4, admite-se que a base de dados, assim como a API implementadas pelo *backend* tem capacidade para armazenar a informação extra. A tabela *fw_core_intl* permite armazenar atributos que não correspondem à estrutura da tabela *fd_core*, de forma a que cada túplo de *fw_core_intl* corresponda a um novo atributo do POI, em que este fica definido pela combinação do identificador do POI, na coluna *uuid*, a chave do novo atributo, na coluna *property_name*, e a linguagem da informação textual, na coluna *lang*. O valor do atributo fica armazenado na coluna *value*. Desta forma, é possível armazenar qualquer número de atributos extra ao POI, eliminado-se a utilização da tabela *poi_extra_data*, cuja estrutura teria de ser refeita, de cada vez que seja implementado um novo tipo de POI no sistema. Elimina-se também a necessidade de manipular a estrutura do POI recebido no servidor, antes de se invocar o *script* de inserção de POIs do *backend* POI-DP. Isto significa que o *script my_add_poi* deixaria de ter utilidade, no entanto, *my_add_poi_segment*, o qual faz parte da criação de segmentos, teria de ser remodelado, visto que se mantém a necessidade de retornar dados do POI recebido para outro *script*.

Embora a criação de segmentos se encontre atualmente operacional, esta funcionalidade é composta por um conjunto significante de operações realizadas a nível do servidor, que poderão ser executadas a nível cliente, poupando-se, desta forma, recursos do servidor. Propõe-se assim, uma nova abordagem de implementação desta funcionalidade, em que a geometria da linha correspondente ao segmento a criar, é gerada no cliente, com recurso às funções da biblioteca *Leaflet*. Em vez de serem submetidos dois POIs para o servidor, a partir dos quais são determinados os *track_points* das extremidades do segmento, e respetiva construção da geometria do mesmo, a estratégia proposta passa por efetuar um pedido ao servidor, após a seleção do percurso onde se pretende criar o segmento, retornando todos os *track_points* referentes a esse mesmo percurso, que são representados graficamente no cliente. O utilizador pode, desta forma, selecionar dois destes *track_points*, entendendo-se que o segmento é definido por todos os *track_points* compreendidos entre os selecionados. A classe *Polyline*, definida pela biblioteca *Leaflet*, permite construir o objeto geográfico referente ao segmento, assim como converter o mesmo em objeto GeoJSON, que pode ser enviado para o servidor, e inserido na base de dados. Esta abordagem permite, em primeiro lugar, descartar o recurso ao *backend* POI-DP, que, de acordo com a presente arquitetura do sistema, encontra-se localizado numa máquina diferente, o que, à partida, tornará o sistema mais flexível em termos

de tolerância a falhas. Em segundo lugar, coloca-se a questão do desempenho. No fundo, esta estratégia transfere carga de processamento do servidor para o cliente, para a execução desta funcionalidade. Por um lado, o servidor deverá ter maior disponibilidade de recursos para suportar múltiplos pedidos. Por outro lado, a aplicação cliente consumirá mais recursos do dispositivo, para executar as operações que, de acordo com a corrente implementação, são efetuadas pelo servidor. Para além desta nova estratégia de implementação, propõe-se também como trabalho futuro, a avaliação do desempenho de ambas as implementações, por forma a determinar as vantagens e desvantagens de cada.

Conforme mencionado previamente, no presente documento, A atual implementação da pesquisa corresponde a uma solução provisória, entendendo-se ser ineficiente. A implementação de *queries* estáticas no servidor para cada combinação possível de parâmetros de pesquisa traduz-se num número demasiadamente elevado de *queries* distintas a implementar, número este que iria escalar com a implementação de novos tipos de informação de valor acrescentados, para além dos que se encontram implementados atualmente. A estratégia proposta passa pela criação de um formulário de pesquisa, o qual permitirá definir uma combinação de parâmetros que, após ser interpretada, permitirá construir dinamicamente, no servidor, a *query* correspondente aos parâmetros definidos.

6.3.2 Funcionalidades Propostas

A determinação de trajetos otimizados implica a implementação de uma funcionalidade de *routing*, assim como a criação de uma topologia. O SGBD utilizado nesta aplicação proporciona as extensões *PostGIS Topology* e *pgRouting*, que permitem criar e manipular topologias, e efetuar operações de *routing*, respetivamente. Propõe-se como alternativa de implementação para esta funcionalidade, a criação de uma topologia, gerada a partir de toda a informação referente às vias rodoviárias, extraídas da cartografia OSM, conjuntamente com os percursos existentes na base de dados. A determinação do caminho será realizada com base nesta topologia em conjunto com um fator de peso, como por exemplo, a velocidade média. Para lidar com o possível problema de certos percursos não serem adequados para certos tipos de veículos, a extensão *pgRouting* permite determinar caminhos alternativos, excluindo-se, desta forma percursos não adequados ao veículo. Propõe-se também que o resultado da pesquisa de caminho possa ser descarregado, segundo um formato padrão, para que o utilizador o possa utilizar para diversas finalidades, entre as quais, navegação.

A navegação corresponde a mais um exemplo de uma funcionalidade a implementar na aplicação móvel proposta anteriormente. A implementação proposta para esta funcionalidade, é relativamente simples. A extensão *geolocation* da biblioteca *JQuery* permite rastrear continuamente a posição do dispositivo. A referida posição pode ser representada sobre a cartografia base com recurso à biblioteca *Leaflet*, assim como a rota que se pretende seguir que deverá ser carregada a partir de uma pasta local. A referida rota corresponderia naturalmente ao resultado obtido da utilização da funcionalidade descrita no parágrafo anterior. Resumidamente, esta proposta de implementação permite ao utilizador visualizar continuamente a posição do seu dispositivo, em relação à rota que que pretende seguir.

A criação de perfis de utilizador fica também em aberto. A existência de utilizadores registados no sistema é pertinente, pois o estabelecimento de uma relação entre os mesmos e os dados inseridos por estes permite acrescentar rastreabilidade aos dados. Para além disto, acrescente ainda suporte para as duas restantes funcionalidades em aberto, nomeadamente a validação de percursos através de uma funcionalidade de *rating*, e divulgação de eventos dos clubes TTT. Propõe-se a implementação de dois perfis de utilizador, o utilizador individual, e o clube TTT. Não é pretendido que as funcionalidades da aplicação se restrinjam a utilizadores não registados, exceto no caso da publicação de eventos TTT. Pretende-se portanto que um

utilizador autenticado possa associar alguns dados pessoais aos percursos e POIs que insere no sistema, proporcionando a referida rastreabilidade da informação, e abrindo caminho à implementação da funcionalidade de *rating*.

A funcionalidade de *rating*, destina-se a cobrir a questão de validação de dados inseridos. Ainda terá de ser estudada a melhor forma de implementação desta funcionalidade, no entanto, pretende-se que os utilizadores possam atribuir uma classificação aos percursos e POIs inseridos, com base na experiência que estes lhe proporcionem, assim como dos próprios utilizadores que efetuaram a inserção dos mesmos dados, com base na qualidade da informação por estes inserida. Esta funcionalidade permitirá à partida mais uma possibilidade de filtragem dos resultados da pesquisa de percursos, permitindo a ordenação dos mesmos com base nas melhores classificações.

A publicação de eventos TTT corresponde a uma funcionalidade exclusiva dos clubes TTT, que poderão aceder a esta funcionalidade, uma vez autenticados no sistema. O *backend* POI-DP, solução já utilizada para a gestão de POIs, poderá ser também uma solução viável para a gestão deste tipo de informação, visto que permite a disponibilização da localização geográfica do evento, em conjunto com toda a sua informação associada. Uma vez inseridos os eventos, estes poderão ser vistos por todos os utilizadores. Propõe-se que, ao iniciar a aplicação, esta disponibilize imediatamente informações sobre um evento, selecionado automaticamente de acordo com a sua proximidade cronológica em relação à data corrente, ou com a sua proximidade geográfica relativamente à posição do dispositivo, ou mesmo ambos os parâmetros. Implicitamente, fica também em aberto, a pesquisa de eventos, que deverá permitir a todos os utilizadores pesquisar todos os eventos inseridos, de acordo com uma combinação personalizada de parâmetros, semelhante à da funcionalidade pesquisa de percursos.

Bibliografia

[1] FIWARE. Fiware.openspecification.miwi.poidataprovider. acedido em 06/02/2015.

[2] FIWARE. File:poi service architecture v3.png. acedido em 06/02/2015.

[3] Markus Neteler and Helena Mitasova. *Open source GIS: a GRASS GIS approach*, volume 689. Springer Science & Business Media, 2013.

[4] Joana Raimundo Oliveira. Análise comparativa de métodos de transformação de coordenadas para a interoperabilidade entre data geodésicos da cartografia do igeoe. 2015.

[5] J Gonçalves. Adopção de sistemas de referenciação geográfica globais. In *Proceedings da conferência ESIG2008, Lisboa*, 2008.

[6] Nathan R Swain, Kilisimasi Latu, Scott D Christensen, Norman L Jones, E James Nelson, Daniel P Ames, and Gustavious P Williams. A review of open source software solutions for developing water resources web applications. *Environmental Modelling & Software*, 67:108–117, 2015.

[7] Guangshi Li. Research on building the information sharing platform based on the ogc web services. In *Information Management, Innovation Management and Industrial Engineering (ICIII), 2013 6th International Conference on*, volume 2, pages 200–203. IEEE, 2013.

[8] Pasquale Di Giovanni, Michela Bertolotto, G Vitiello, and M Sebillo. Web services composition and geographic information. *Geographical Information Systems: Trends and Technologies*, pages 104–141, 2014.

[9] Gérald Fenoy, Nicolas Bozon, and Venkatesh Raghavan. Zoo-project: the open wps platform. *Applied Geomatics*, 5(1):19–24, 2013.

[10] Opengis web map tile service implementation standard. Technical Report 07-057r7, OGC - Open Geospatial Consortium, 2010.

[11] OGC Open Geospatial Consortium. Opengis catalogue services specification. Technical Report 07, OGC - Open Geospatial Consortium, 2007.

[12] Simone Giannecchini. Geoserver, the open source server for interoperable spatial data hand-ling. *Untersuchung der Nutzung von OpenStreetMap Daten zur Darstellung von TMC Verkehrsmeldeinformatio*, 2013.

[13] Andrea Ballatore, Ali Tahir, Gavin McArdle, and Michela Bertolotto. A comparison of open source geospatial technologies for web mapping. *International Journal of Web Engineering and Technology*, 6(4):354–374, 2011.

[14] Richard G Donohue, Carl M Sack, and Robert E Roth. Time series proportional symbol maps with leaflet and jquery. *Cartographic Perspectives*, (76):43–66, 2014.

[15] GDAL/OGR. Gdal. acedido em 28/10/2015.

[16] Frank Warmerdam. The geospatial data abstraction library. In *Open Source Approaches in Spatial Data Handling*, pages 87–104. Springer, 2008.

[17] Gerald I. Evenden. Cartographic projection procedures for the unix environment - a user's manual. Technical Report 90, United States Department of the Interior, 1990.

[18] OSGeo. Geos. acedido em 28/10/2015.

[19] FIWARE. Overall fiware vision. acedido em 07/11/2015.

[20] FIWARE. Poi data provider open api specification. acedido em 14/10/2015.

[21] wiki.openstreetmap.org. Bounding box. acedido em 06/11/2015.

[22] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.

[23] Pascal Neis and Alexander Zipf. Analyzing the contributor activity of a volunteered geographic information project—the case of openstreetmap. *ISPRS International Journal of Geo-Information*, 1(2):146–165, 2012.

[24] Mordechai Haklay et al. How good is volunteered geographical information? a comparative study of openstreetmap and ordnance survey datasets. *Environment and planning. B, Planning & design*, 37(4):682, 2010.

[25] OSM. Tile.openstreetmap.org/usage policy. acedido em 02/10/2015.

[26] SWITCH2OSM. Manually building a tile server (14.04). acedido em 06/02/2015.

[27] wiki.openstreetmap.org. File:osm components.png. acedido em 06/11/2015.

[28] Nikolay Teslya. Web mapping service for mobile tourist guide. In *Open Innovations Association FRUCT, Proceedings of 15th Conference of*, pages 135–143. IEEE, 2014.

[29] wiki.openstreetmap.org. Osm2pgsql. acedido em 06/11/2015.

[30] wiki.openstreetmap.org. Mapnik. acedido em 06/11/2015.

[31] wiki.openstreetmap.org. mod_tile. acedido em 06/11/2015.

[32] Jason G Su, Meghan Winters, Melissa Nunes, and Michael Brauer. Designing a route planner to facilitate and promote cycling in metro vancouver, canada. *Transportation research part A: policy and practice*, 44(7):495–505, 2010.

[33] Hartwig H Hochmair and Claus Rinner. Investigating the need for eliminatory constraints in the user interface of bicycle route planners. In *Spatial Information Theory*, pages 49–66. Springer, 2005.

[34] Harald Holone, Gunnar Misund, and Hakon Holmstedt. Users are doing it for themselves: Pedestrian navigation with user generated content. In *Next Generation Mobile Applications, Services and Technologies, 2007. NGMAST'07. The 2007 International Conference on*, pages 91–99. IEEE, 2007.

[35] Julia Letchner, John Krumm, and Eric Horvitz. Trip router with individualized preferences (trip): Incorporating personalization into route planning. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 1795. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[36] Roberto Osegueda, Alberto Garcia-Diaz, Suleiman Ashur, Octavio Melchor, Sung-Ho Chang, Cesar Carrasco, and Ahmet Kuyumcu. Gis-based network routing procedures for overweight and oversized vehicles. *Journal of Transportation Engineering*, 125(4):324–331, 1999.

[37] Ravishanker Kusuma. jquery upload file plugin demo. acedido em 12/10/2015.

[38] FIWARE. Poi data provider - user and programmers guide. acedido em 06/02/2015.

[39] Dan Robinson. Postgresql's powerful new join type: Lateral. acedido em 16/11/2015.

[40] Regina O Obe and Leo S Hsu. *PostgreSQL: Up and Running: A Practical Introduction to the Advanced Open Source Database*. "O'Reilly Media, Inc.", 2014.

[41] GEOFABRIK. http://download.geofabrik.de/europe/portugal.html. acedido em 06/11/2015.

[42] Open Source Geospatial Foundation. Wfs vendor parameters. acedido em 17/11/2015.

[43] Rohit Ghatol and Yogesh Patel. Beginning phonegap. *New York: Apress Media*, 2012.